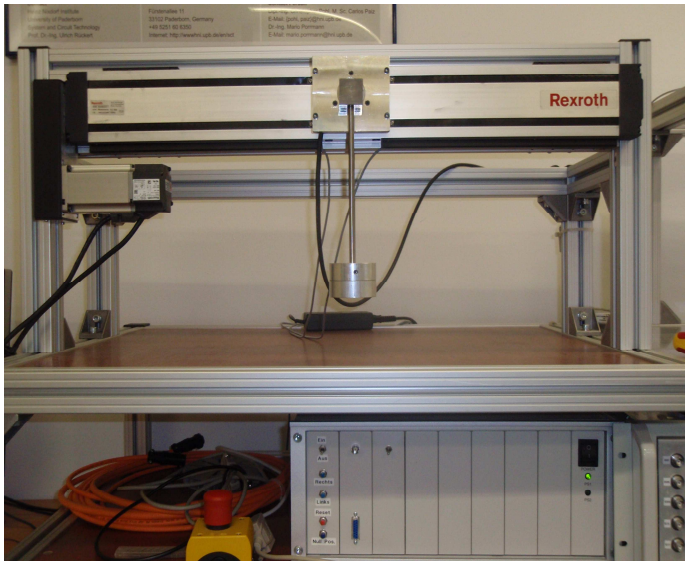


FPGA-Based Model of an Inverted Pendulum Hardware-in-the-Loop Simulations

Roberto Gregorio Moreno

Tutor: Carlos Paiz Gatica

FPGA-Based Model of an Inverted Pendulum for Hardware-in-the-Loop Simulations

ABSTRACT

An FPGA-Based Model of an Inverted Pendulum for Hardware-in-the-Loop Simulations consists of a specific study of the SCT Pendulum dynamics in order to find a realistic model and its later implementation in FPGA technology. The design flow is based on Matlab Simulink/Xilinx System Generator, final results having proven this hardware model to be a good substitute for the real pendulum, which makes it an ideal tool for testing controllers in Real-Time and Hardware-in-the-Loop Simulations.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER II

CHAPTER III

CHAPTER IV

LIST OF TABLES

0. ACKNOWLEGDMENTS

The first thing is to thank everyone who helped me in the realization of this master thesis. Starting with my tutor Carlos Paiz, whose help and patience were fundamental; continuing with Shaady Khatab and Martin Krüger, thank you very much for giving me light in the darkness of mechatronics; also to everybody of the System and Circuit Technology (STC) research group, headed by Prof. Dr Ing. Ulrich Rückert, which gave me this great opportunity to do my master thesis at the Heinz Nixdorf Institute of the University of Paderborn (Germany); I can not forget my friends and colleagues working many pizza-nights in the HNI, Christian Martin and Risang G. Yudanto; and to all the people who heard me and gave me their support, sharing a Paderborner, whenever I needed someone to encourage me on: Rafa, Pablo, Crespi, Eder, Bea, Anita, Laura... and how could I forget everybody who came to my master thesis presentation: Dani, Sohrab, Elena, Laura (you did not come but you were there in spirit) and particularly to Patri, I have no words to thank all you did for me.

Finally, this project is dedicated to my parents, for all the support and all the suffering that they have passed through my education, and, especially, this master thesis is dedicated to my grandmother, who has been seen a dream come true, having a grandson as telecommunications engineer.

Thank you very much everybody.

1. INTRODUCTION

This master thesis is based on the pendulum used in STC department of University of Paderborn. This pendulum is a testing bench for control algorithms. The objective is to balance the bob of the pendulum when it is above the pivot point and that is why it is called inverted pendulum. This position is unstable but using a feedback system and controller devices it is possible to balance the pendulum.

The inverted pendulum has no sense as a product, but is much extended in electronic departments because it is a simple and well-known dynamic system for testing control techniques, which are the real products. Some commercial applications for these controllers are rocket guidance, self-balancing personal transporter, lifting cranes on shipyards.

At the STC department the inverted pendulum is used to test reconfigurable hardware to implement digital controllers using FPGA-technology. FPGAs are integrated in a motherboard that is called RAPTOR system. The RAPTOR system is a modular FPGA-based rapid prototyping, developed by SCT department.

The testing bench consists on connecting the outputs of the inverted pendulum to the RAPTOR 2000 by its PCI- Bus interface, then through a Local Bus these signals are sent to the Module where is placed a Virtex II Pro FPGA. There, the controllers calculate the next set point for the pendulum and send the voltage signal corresponding with this goal point. Like that is how the feedback loop is built.

Reconfigurable hardware consist on swapping between swing up controller circuit, which has the mission to bring the pendulum to the upper position by swinging, and balance controller circuit, which has the aim of maintaining balance and equilibrium of the pendulum in this inverted position. For sure, this swapping between these hardware controller circuits at the FPGA must be fast enough because the system works in real time.

The benefits of using this dynamically reconfigurable hardware instead of two FPGA with a controller in each, or the reasons why it is better using FPGAs than micro-controllers will not be discussed in this master thesis. To know more about it we recommend [1], [2], [3] and [4]

There is an important fact that has to be considered: it is necessary a good and realistic model, or plant, of the pendulum for designing good controllers. Prior to the realization of this master thesis, the controllers were designed from a very basic and inaccurate pendulum model and even if these controllers work can be improved.

Finally, we want to introduce the term Augmented Reality. It is a technology that provides a direct or indirect view where there are real and virtual elements mixed. It is an interesting concept for us because the hardware pendulum model is implemented in a chip; therefore, the “magic” view of an inverted and balanced pendulum will replace the boring graphs. The idea is to show at the screen a virtual image of a pendulum, which will move in real time as the same way that the hardware model will indicate.

Using Augmented Reality the hardware model of the pendulum will have the perfect partner. Both together will be an accurate and virtual replacement for the real pendulum.

After this short introduction we are ready to show the purpose of this master thesis

1.1. Purpose

Find out a realistic model of the SCT pendulum and implement it in hardware using a FPGA at a module of the RAPTOR system. The goal is to substitute the real pendulum for this accurate model making it an ideal and, especially, small and handy tool to test controllers in Real-Time and Hardware-in-the-Loop Simulations. And finally, test the compatibility between the hardware model with the Augmented Reality in real time.

1.2. Objectives

There are many benefits for the department SCT as a result of the realization of this master thesis, for instance:

- The hardware model may replace the real pendulum, making the system much smaller and manageable. It will allow testing in real time the controllers without using the real pendulum, which weighs over 30 Kg. and occupies too much space; thereby, these tests could be done on different sites to the SCT laboratory, even at home.

- Real time tests will be faster, cheaper and safer than using real pendulum because there will not be real elements as Power Driver or AC Motor, which requires time (turn on, setup), energy (high current is needed by motor) and safety measures (to avoid risks such as high current or dangerous movements from pendulum) for working fine.

- Dangerous tests, which can damage the real system, could be done because the hardware model will never break its motor, cart or spindle.

- This better model of the inverted pendulum system will provide better controller designs. Remember that every controller design is based on dynamic equations of the system to control and after this master thesis the equations obtained are more accurate than previously. Therefore, the future controllers will be better.

1.3 Thesis overview and structure

This master thesis is structured in three main parts. Now a brief overview of each part is shown:

- Modelling the SCT pendulum: This section begins with a short description of the system, its elements and features, including a brief explanation of the behaviour of the SCT pendulum system. After this master thesis continues modelling the system by blocks. There are three main block, the first one correspond to the mechanic block, which has an input voltage provides by the controllers block and a force output, which will move the pendulum; the second one is the pendulum block, which will be modelled using differential equations and applying accurate models of friction; and the last one is the encoders block, which converts the outputs of the pendulum in TTL signals, which will be the inputs of the controllers block and they are necessaries for calculating the next voltage set point for the

mechanic block closing the feedback loop. Finally, this section finishes giving values to every parameters found previously, by means of calculation or identification after some experiments with the real pendulum.

- Model representation on Simulink: This section is focus on finding an equivalent schematic of the model using Simulink software of Matlab. In this section there is a comparison between our model and the actual pendulum using real data from an experiment where the controllers achieve the stabilization of the pendulum.

- Hardware description of the model: This section is based on converting the software model of the pendulum using Simulink blocks in a hardware model which will be implemented in the FPGA. The first part consist on replacing Simulink blocks by Xilinx System Generator blocks, some blocks are equivalents but others need engineering solutions to find out the equivalence. After that the netlist is created, then using V-MAGIC (a software developed by SCT department) a VHDL file is generated, which includes the programming lines that are necessaries to be compatible with the RAPTOR 2000 interface. Finally, using Xilinx ISE project navigator the bitstream is generated and the FPGA will be programmed with our hardware pendulum model. Some tests are done for comparing this hardware model with the software model and the actual pendulum. And the final results are discussed.

2. MODELLING THE PENDULUM

2.1 Pendulum system description: Elements, features and behaviour

In this section, the STC inverted pendulum, which is the system to model and shown in Figure II.1, will be described from three different points of view: their elements, features and behavior.

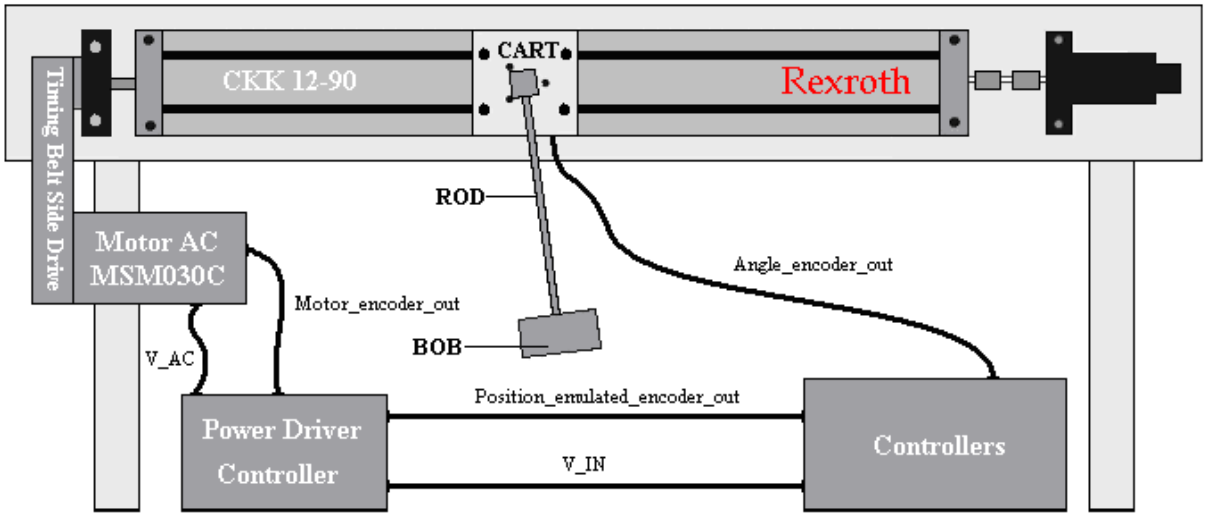


Figure II. 1: SCT pendulum diagram

2.1.1. ELEMENTS

In Figure II.1 are shown the following elements of the pendulum system:

AC servo Motor: Rexroth MSM030C-0300-NN-M0-CG0.
Timing Belt: Gear ratio equal to 1.
Compact Module: Rexroth CKK_12_90 (ball screw drive).
Power Driver Controller: Rexroth ECODRIVE Cs DKC01.3-012.3-FW
Pendulum:

- Cart.
- Rod.
- Bob.

Motor Encoder: Absolute encoder used to find out cart position
Angle Encoder: Heidenhain ROD 420, an incremental rotary encoder used to find out the pendulum angle.

2.1.2. FEATURES

Motor:

MSM030C-0300-NN-M0-CG0 is an AC servo motor without brake providing a plain shaft with 400W of power. It has an absolute multi-turn encoder that is connected to the drive controller by a serial interface (2.5MBaud). The most important parameters for us are the torque constant (K_M) and the rotor inertia (J_M) because these electric and mechanical parameters, respectively, are used to model the pendulum [5].

Timing Belt:

It is the option of attaching the AC servo motor to the Compact Module CKK 12-90 used in the SCT pendulum. There are different gear ratios available; in our case the rate used is 1:1, which means that one revolution in the servo motor is equivalent to one revolution of the spindle inside the Compact Module.

Compact Module:

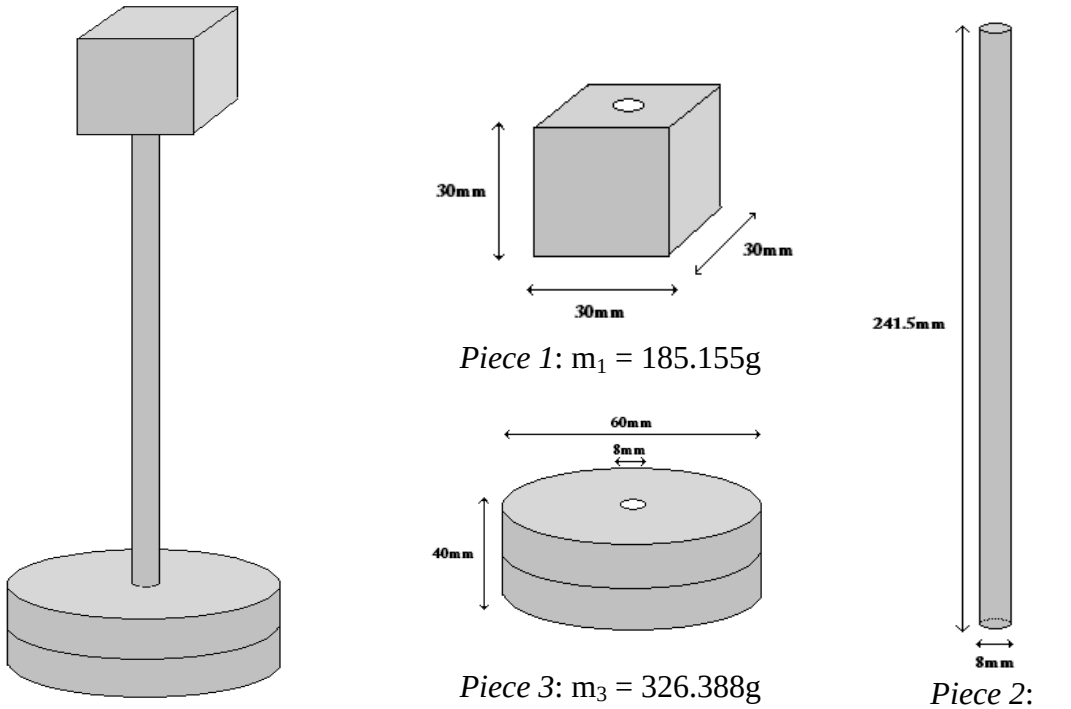
CKK 12-90 is a 750mm length Compact Module consisting of a guideway, which is a ball rail system, and a drive unit, which is a ball screw drive. The numbers are the guideway (12) and the frame (90) dimension in millimetres. The most important feature is the feed constant, which indicates how long the cart moves forward per spindle revolution. This constant is called k in the driver software (DriveTop16V14) and its value is 5mm/rev. It is possible to obtain this value by checking the Compact Module datasheet[6] for the dimensions of the ball screw (d_0xP = 12x5 mm).

Power Driver Controller:

There is an interesting parameter in order to model the SCT pendulum in the power driver controller: the resolution of the incremental encoder emulation for position of the cart, which is 500 inc/rev (rotary resolution) or 10µm/inc (linear resolution) if the feed constant is used to convert rotary resolution into linear resolution (cart position).

Pendulum:

- Cart: mass is the only important feature in the cart: M_{cart} = 0,5146Kg.
- Rod & Bob: mass and dimensions are important for modelling as they are necessary to calculate the centre of mass and the moment of inertia of the pendulum. $M_{rod\&bob}$ = 0,581Kg. Figure II.2 reflects the dimensions and masses measured using a precision scale and caliber.



Piece 1: m_1 = 185.155g

Piece 3: m_3 = 326.388g

Piece 2: m_2 = 69.457g

Figure II. 2: Rod & Bob diagram: Pieces masses and dimensions

Motor Encoder:

This absolute rotary encoder that has a resolution of 131072 inc/rev sends the value to the power driver controller, which emulates it as an incremental encoder sending the value of the position of the cart as TTL format to the controller block.

Angle Encoder:

The Heidenhain ROD 420 incremental rotary encoder has a resolution from 50 to 5000 lines/rev; particularly, the encoder of the SCT pendulum has a resolution of 5000 lines/rev. The output is sent directly to the controller block as TTL format.

2.1.3. BEHAVIOUR

In order to understand the behaviour of the SCT pendulum system and be able to model it in a simple way, six subsystems have been created and illustrated on the diagram below:

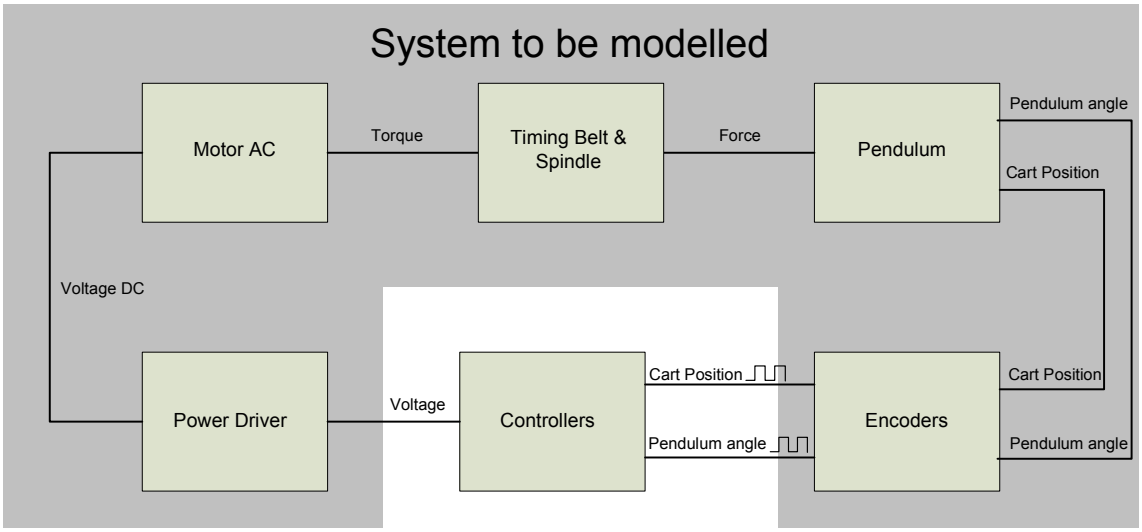


Figure II. 3: Blocks diagram of the STC Pendulum System to be modelled

As it can be seen in Figure II.3, all the blocks, but the controllers, belong to the system to be modelled in this master thesis. To simplify the diagram, the encoders have been grouped in one block, which takes its inputs directly from the pendulum and connects its TTL outputs to the controllers block. In the actual pendulum, as seen in figure 1, the absolute encoder of the position is situated in the motor, from which it is connected to the power driver controller, which takes the position value and emulates an incremental encoder providing controllers with a TTL signal.

Therefore, and from this point on, the cart position encoder will be considered as an incremental rotary encoder which takes its input from the output of the pendulum generating a TTL output with a resolution of 500 inc/rev.

In order to explain the behaviour of the SCT pendulum system we will start from the controllers block. As stated in the introduction, the controllers determine the next set point for the cart, either swinging up or balancing the pendulum. Therefore this block sends the corresponding voltage signal to the power driver, which converts this signal in a valid input signal (appropriate voltages and times) for the motor.

Since the moment the motor receives its input signal it converts it into a torque and transmits it to the spindle using the timing belt. This torque transmitted to the spindle corresponds to the motor torque because the gear ratio in the timing belt side drive is 1:1. Then the ball screw drive is moved by the rotary movement of the spindle and finally the cart is moved to the right or left depending on the spin of spindle: clockwise or counterclockwise respectively. Consequently, this cart movement makes the rod and bob move too, modifying the value of the pendulum angle or its angular position.

The angular and cart position, which are the outputs of the pendulum, are coded by the incremental encoders block in TTL format. These signals are sent to the controllers block, where this information is needed to calculate the new set point. Then this block generates and sends the new corresponding voltage signal as input to the power driver, starting again the loop.

As seen, this is basically the behaviour of the SCT pendulum system. Although some blocks might be explained more in-depth, which will be done in following sections, the goal of this section was to show how the system works and to simplify the way to model it.

2.2. Modelling Power Driver Controller, Motor AC, Timing Belt and Spindle

The goal of this section is to model some blocks of the pendulum system, which are Power Driver Controller, Motor AC, Timing Belt and Spindle block, in the easiest way but, obviously, also realistically and correctly. As is shown in Figure II.3, the output of Timing Belt and Spindle block is a force, which is the input for Pendulum block, it is important that this force includes all the mechanical effects (like the moment of inertia of motor and spindle).

As it can be seen in Figure II.4, where appear the first three consecutive blocks from Controllers block shown in Figure II.3, these three blocks can be modelled as a unique block that converts the input voltage having a range from 0 to 10V into a signed force output which is proportional to the input.

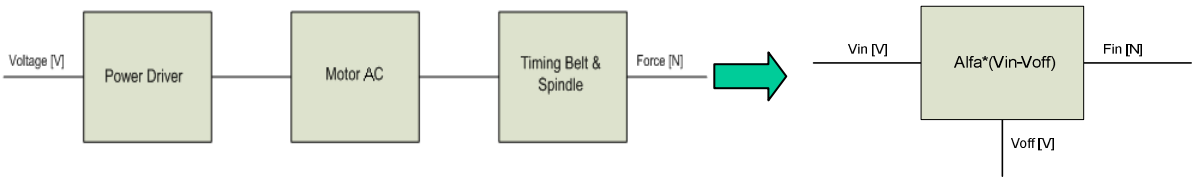


Figure II. 4: Blocks diagram. Power Driver, Motor and Timing Belt & Spindle are modelled as a block

In order to make this simplification, a linear model, it is necessary to understand how the Power Driver Controller works. The analog input voltage is converted into digital voltage by the ADC, then the voltage offset of 5V is subtracted to obtain a new range [-5V, 5V]. Finally it is multiplied by the scaling factor, which is 150% per 10V. This Power Driver Controller behaviour is shown in Figure II.5, which is a screenshot from the driver start-up software (DriveTop16V14 [7]).

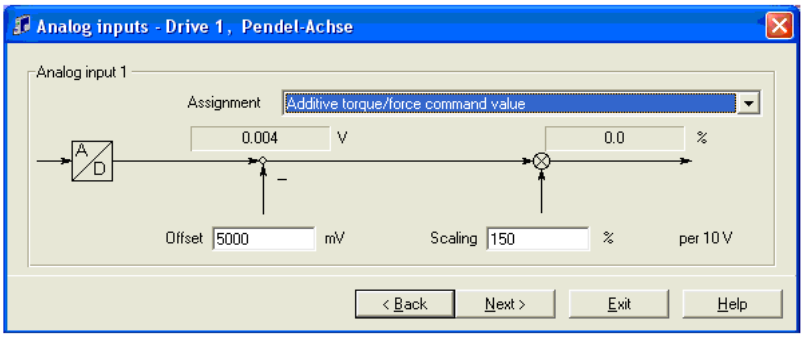


Figure II. 5: Power Driver behaviour

The next step is to model the AC Motor, where the input voltage is converted into a proportional torque output. This conversion is done by the torque constant (K_M), multiplying the maximum active peak current (I_{Active_Peak}) provided by the Power Driver Controller. The torque constant can be found in the motor datasheet[5] and the maximum active peak current can be found in the Drive limitation window from the driver software (DriveTop16V14[7]), whose screenshot is shown in Figure II.6.

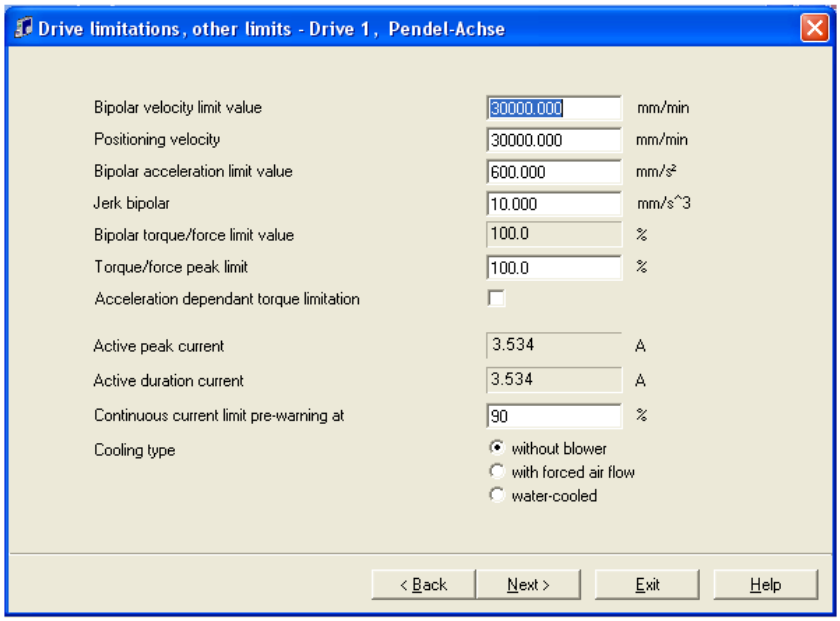


Figure II. 6: Maximum active peak current: 3.534 [A]

It has to be noted that there is a continuous current limit pre-warning that will be used in the formula.

The final step is converting this torque into force, which is done by the Spindle block, where the feed constant *k* indicates how many meters the cart moves per revolution of the Spindle. Therefore, the output force from this model is represented as F_{IN} because it is the input force in the pendulum block and is obtained by the following formulas:

$$F_{IN} = \frac{150\%}{10V}(V_{IN} - V_{OFF}) \cdot I_{Active\_Peak} \cdot 90\% \cdot K_m \cdot \frac{1}{k} = \alpha \cdot (V_{IN} - V_{OFF}) \quad \text{(II.1)}$$

Where:

F_{IN} : Input force in the pendulum block [N]

V_{IN} : Input voltage in the power driver block [V]

V_{OFF} : Offset voltage [V] = $5V$

I_{Active_Peak} : Max. active peak current [A] = $3.534A$

K_m : Torque constant of motor $\left[\frac{N \cdot m}{A}\right] = 0.54 \frac{N \cdot m}{A}$

k : Feed constant of spindle block $[m] = 5 \cdot 10^{-3} m$

α : Voltage to force conversion factor $\left[\frac{N}{V}\right]$

Finally, the value of α is:

$$\alpha = \frac{150\%}{10V} \cdot I_{Active\_Peak} \cdot 90\% \cdot K_m \cdot \frac{1}{k} = 51.525 \left[\frac{N}{V}\right] \quad \text{(II.2)}$$

Before concluding this section, it is important to emphasize that there are two moments of inertia that cannot be neglected: Moment of inertia from the Motor (J_M) and Spindle (J_{SP}).

The Figure II.7 shows how the input voltage (V_{IN}) is converted to torque (M_A) and how this torque, after interacting with the moment of inertia of Motor (J_M) and Spindle (J_{SP}), moves the cart a distance of X_K.

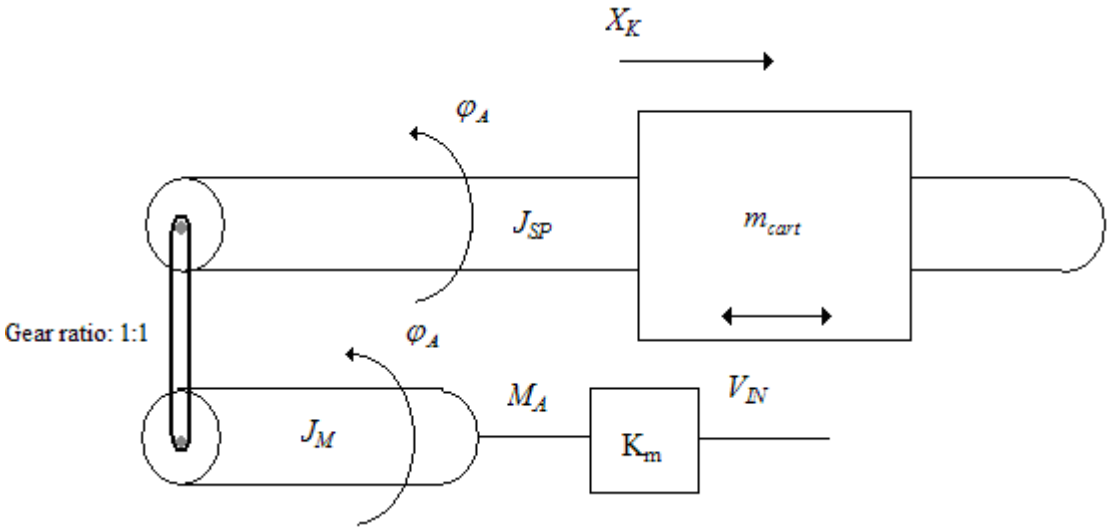


Figure II. 7: Mechanical diagram that explains how input voltage moves the cart

The equilibrium condition in this system is given by:

$$\left(J_{SP} + J_M\right) \cdot \ddot{\varphi}_A + m_{cart} \frac{\ddot{X}_K}{i} = M_A \quad \text{(II.3)}$$

$$\text{Where } i = \frac{2\pi}{k}\left[\frac{rad/rev}{m/rev}\right] = \frac{2\pi}{5e^{-3}}\left[rad/rev\right] = 400\pi\left[rad/rev\right] \quad \text{(II.4)}$$

As $F_A = M_A \cdot i$ and $\varphi_A = X_K \cdot i$ it is possible to write the equilibrium condition as a force equation instead of a torque equation:

$$\left(J_{SP} + J_M\right)\cdot \ddot{X}_K \cdot i + m_{cart} \cdot \frac{\ddot{X}_K}{i} = \frac{F_A}{i} \rightarrow \left(J_{SP} + J_M\right)\cdot \ddot{X}_K \cdot i^2 + m_{cart} \cdot \ddot{X}_K = F_A \rightarrow$$
$$\rightarrow \left(\left(J_{SP} + J_M\right)\cdot i^2 + m_{cart}\right)\cdot \ddot{X}_K = F_A \rightarrow M = \left(\left(J_{SP} + J_M\right)\cdot i^2 + m_{cart}\right)[Kg] \text{ (II.5)}$$

The mass M is a "virtual" mass of the cart because it includes the moment of inertia of both Motor and Spindle. Its value will be much greater than the value of mass of the cart measured on the scale (remember features of pendulum section) and to obtain M it is necessary to calculate J_{SP} because J_M can be found in the Motor datasheet [5].

J_{SP} is denoted in the page 29 of the Compact Module datasheet [6] as J_{fr} and its formula, for motor attachment via timing belt side drive, is:

$$J_{fr} = (k_1 + k_2 L + k_3 m_{fr})\cdot 10^{-6} + J_{Rv} + J_{Br} \text{ (II.6)}$$

Where L is the length of the Compact Module given in millimetres, L = 750mm; m_{fr} is the mass of external load given in kilograms, $m_{fr} = M_{cart} + M_{rod\&bob}$ = 1.096Kg; J_{Rv} is the reduced mass moment of inertia of timing belt side drive at motor journal given in kilograms times square meter, $J_{Rv} = 38e^{-6}\left[Kg \cdot m^2\right]$ as is shown in the CKK 12-90 datasheet [6]; the k_1, k_2, k_3 are constants whose values are defined in the table of Figure II.8. See second row (12x5) and 1 carriage column:

Size	Ball screw	Constants k_1		k_2	k_3	Frictional torque M_{RS}
	d_0 x P	1 carriage	2 carriages			(Nm)
CKK 12-90	12 x 2	1.279	1.303	0.013	0.101	0.11
	12 x 5	1.454	1.600	0.011	0.633	0.15
	12 x 10	2.138	2.750	0.011	2.533	0.18

Figure II. 8: Table of datasheet for CKK 12-90 [6] where constants k_1, k_2 and k_3 are defined

Note that J_{Br} = 0, because it is the mass moment of inertia of motor brake and this is a system without break.

$$J_{fr} = (k_1 + k_2 L + k_3 m_{fr})\cdot 10^{-6} + J_{Rv} + J_{Br} =$$
$$= (1.454 + 0.011\cdot 750 + 0.633\cdot 1.096)\cdot 10^{-6} + 38\cdot 10^{-6} + 0 = 48.4\cdot 10^{-6}\left[Kg\cdot m^2\right] \text{ (II.7)}$$

Therefore, taking the value of J_M from the datasheet [5], the mass M is finally obtained as:

$$M = \left(J_{SP} + J_M\right)\cdot i^2 + m_{cart} = \left(48.4 + 17\right)\cdot 10^{-6}\cdot (400\pi)^2 + 0.5146 = 103.205[Kg] \text{ (II.8)}$$

This is all the modelling process from the previous blocks to the pendulum block, which will be the next target to be modelled. The following section will be dealing with it.

2.3. Modelling the Pendulum by using differential equations

Several methods can be used to model the Pendulum block, for instance a neural network or multi-body simulation software as SimPack[8], but the analytic method was chosen because it was the simplest and as precise as necessary due to dynamics is well known for us and the other methods require extra knowledge even learning to use an extra software. Remember that one of the objectives of this master thesis is to find a very realistic and precise model of the real SCT pendulum.

Focusing on the analytic method, the next decision to take was which mechanism to choose in order to represent the pendulum dynamics. Remembering the classical mechanic we always think about Newton's equations of motion, but there are two alternative formulations of classical mechanics: Lagrangian and Hamiltonian mechanics.

These methods had arisen from Newton Laws; therefore, they are more abstract and general than the Newtonian method. The main difference is that they use the energy and work (scalar magnitudes) instead of force (vectorial magnitude) for describing a mechanical system. In simple cases Newtonian method is perfect, however when the mechanical system is more complex, such as a pendulum included in a moving cart, this method becomes complicated.

The main advantage of Lagrangian and Hamiltonian method against Netwon is that they use generalized coordinates, which allow to obtain invariant equations of motion independently of changes of references because the generalized coordinate values do not depend on any other coordinate on the plane and it is a great advantage for complex mechanical systems. Thereby, these coordinates describe the motion of the SCT pendulum, which is a free pendulum included in a moving cart, in an easier way than through Newton equations, consequently, Lagrangian or Hamiltonian method will be chosen.

Both methods could be chosen, but the Lagrangian method was selected because the second-order differential equations obtained are easily solved and, therefore, is not necessary to use the Hamiltonian method. Remember that on a n-dimensional coordinate space (where n is the number of degrees of freedom of the system) the Lagragian method gives n second-order differential equations instead of 2n first-order differential equations given by the Hamiltonian method. This mean: if n second-order differential equations are easily solved, then, it is better solving them than solving 2n first-order equations, and this is our case.

As seen in Figure II.9, the generalized coordinates, the horizontal displacement of the cart (x) and the rotational displacement of the pendulum (θ), were defined on the plane XY.

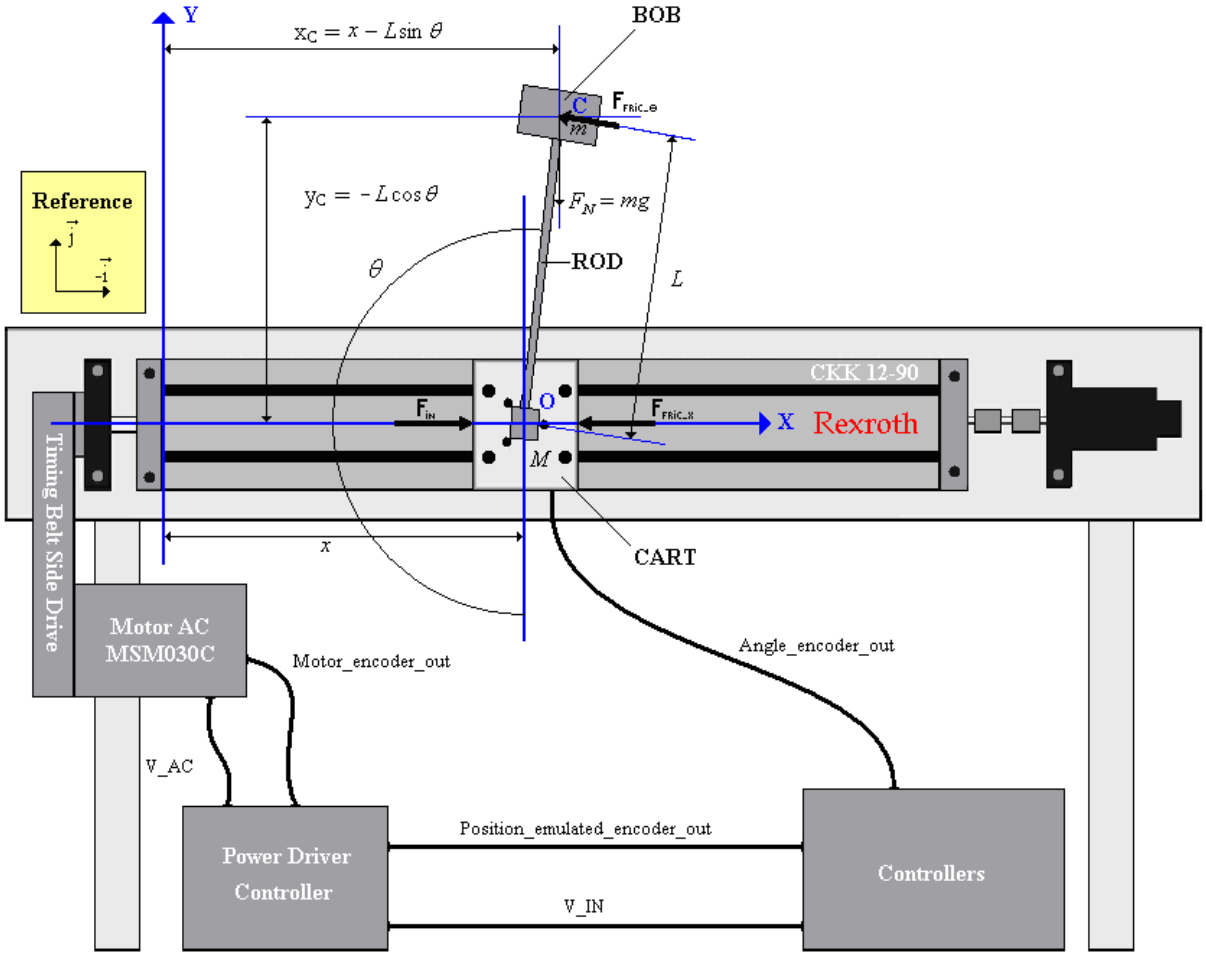


Figure II. 9: SCT pendulum system schematic including: plane XY, its unitary vectors (yellow box), centre of mass of pendulum (point C), its coordinates (x_C, y_C), generalized coordinates (x, θ), masses of pendulum (m) and cart (M), and forces diagram: F_{IN}, input force; F_{FRIC_X}, friction force in the cart; F_N, normal force; and F_{FRIC_θ}, friction force in the pendulum.

2.3.1. Friction forces

This is an important section in our master thesis because for finding out an accurate model of the SCT pendulum the friction forces could not be neglected. Usually, these forces are considered as a simple viscous friction model [9] [10] [11] [12] [13], but it is not accurate enough for us.

We have considered also the static and dynamic (or Coulomb) friction forces in our pendulum model. As is demonstrated in [14] neglecting these forces introduces some inaccuracies.

To illustrate this, the Figure II.10, which is taken from page 3 of [14], is shown. Due to friction forces a ripple is observed in the experiment line, which is shown in Figure II.10 with a solid line.

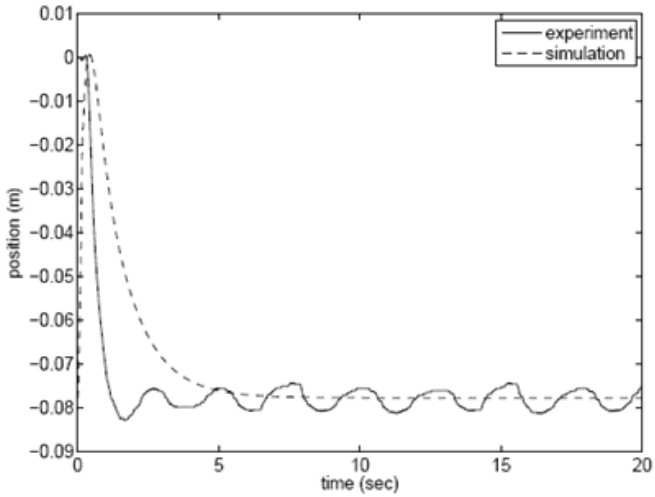


Figure II. 10: Experiment and simulation with simple viscous friction model [14]

Sue Ann Campbell et al. paper [14] focuses on designing better controllers; in our case, this objective is a secondary target because the main target is to obtain an accurate model of the pendulum. Of course, this accurate model could be used to design better SCT pendulum controllers in further works.

At this point, the process of modelling the pendulum block can start. At the end of this section some equations that describe the dynamic of pendulum will be found out. Remember the outputs will be cart position (x) and angle position (θ) and the input of this pendulum block will be the force supplied by the Spindle. The first step of this modelling process is to define the generalized coordinates and forces.

2.3.2. Generalized Coordinate System and Forces

The pendulum system has two degrees of freedom that are represented using two generalized coordinates (ξ_i): the horizontal displacement of the cart (x) and the rotational displacement of the pendulum (θ). Mathematically written as follows:

$$\xi_i : x, \theta \quad \text{(II.9)}$$

The horizontal displacement of the cart has its positive direction to the left and the rotational displacement of the pendulum has its positive direction clockwise, measured from the downward position.

The pendulum system is classified as a holomonic dynamic system because every constraint of the system depends only on the coordinates and time, and not on velocities. Therefore the set of the associated admissible variations are:

$$\delta\xi_i : \delta x, \delta\theta \quad \text{(II.10)}$$

Finally, the generalized forces (Ξ_i) from the non-conservative work are mathematically defined as:

$$\delta W^{nc} = \sum_{i=1}^{N} \Xi_i \delta\xi_i \quad \text{(II.11)}$$

Identifying the non-conservative forces in the SCT pendulum system:

- *f(t)*: input force that comes from the torque of the AC motor and is supplied by the spindle.
- $f_{FRIC\_X}(t)$: friction force in the cart. This force is subdivided in three non-conservative forces too:

  - $f_{STATIC\_X}(t)$: static friction force in the cart, which only appears when the cart's velocity is zero ($\dot{x} = 0$). Its value depends on *f(t)* and F_N is the normal force:

$$f_{STATIC\_X}(t) = \begin{cases} -f(t) \leftrightarrow |f(t)| < \mu_{S\_X} F_N \\ -\mu_{S\_X} F_N \operatorname{sgn}(f(t)) \leftrightarrow |f(t)| \geq \mu_{S\_X} F_N \end{cases} \quad \text{(II.12)}$$

  - $f_{COULOMB\_X}(t)$: Coulomb friction force in the cart, which only appears when the cart is moving ($\dot{x} \neq 0$).

$$f_{COULOMB\_X}(t) = -\mu_{C\_X} F_N \operatorname{sgn}(\dot{x}) \quad \text{(II.13)}$$

  - $f_{VISCOUS\_X}(t)$: viscous friction force in the cart, which, like Coulomb friction force, only appears when the cart is moving ($\dot{x} \neq 0$).

$$f_{VISCOUS\_X}(t) = -b_x \dot{x} \quad \text{(II.14)}$$

- $f_{FRIC\_\theta}(t)$: friction force in the pendulum. This force is also subdivided in three non-conservative forces:

  - $f_{STATIC\_\theta}(t)$: static friction force in the pendulum, which only appears when the pendulum's velocity is zero ($\dot{\theta} = 0$). Its value depends on *f(t)*:

$$f_{STATIC\_\theta}(t) = -\mu_{S\_\theta} F_N sen(\theta) \quad \text{(II.15)}$$

  - $f_{COULOMB\_\theta}(t)$: Coulomb friction force in the pendulum, which only appears when the pendulum is moving ($\dot{\theta} \neq 0$).

$$f_{COULOMB\_\theta}(t) = -\mu_{C\_\theta} F_N sen(\theta) \quad \text{(II.16)}$$

  - $f_{VISCOUS\_\theta}(t)$: viscous friction force in the pendulum, which, like Coulomb friction force, only appears when the pendulum is moving ($\dot{\theta} \neq 0$).

$$f_{VISCOUS\_\theta}(t) = -b_\theta \dot{\theta} \quad \text{(II.17)}$$

After testing experimentally that the friction on the axis of rotation of the pendulum is almost zero ($f_{STATIC\_\theta}(t) \approx 0$), we have to neglect Coulomb friction ($f_{COULOMB\_\theta} \approx 0$) because it is smaller than the static one ($f_{COULOMB\_\theta} < f_{STATIC\_\theta}(t)$). Therefore, only viscous friction force

generated at the pivot, $f_{VISCOUS\_\theta}(t)$ has to be considered in the rotational movement of the pendulum.

Hence the generalized forces (Ξ_i) from the non-conservative work are:

$$\Xi_x = f(t) + f_{FRIC\_X}(t) \quad \text{(II.18)}$$
$$\Xi_\theta = f_{FRIC\_\theta}(t) = f_{VISCOUS\_\theta}(t) = -b_\theta \dot{\theta} \quad \text{(II.19)}$$

2.3.3. Lagrangian

The definition of Lagrangian is:

$L = T - V$ (II.20), where T is the kinetic energy function and V the potential energy function.

We know $T = T_{CART} + T_{PENDULUM}$ (II.21)

$$T_{CART} = \frac{1}{2} M \dot{x}^2 \text{ and } T_{PENDULUM} = \frac{1}{2} m \vec{v}_c \bullet \vec{v}_c + \frac{1}{2} I w^2 \quad \text{(II.22)}$$

Where:

M: mass of cart
m: mass of pendulum (rod + bob)
$\vec{v}_c$: velocity of the pendulum's centre of mass, which is the point named as C in Figure II.9. Remember $\vec{v}_c = \frac{d\vec{r}_c}{dt}$, $\vec{r}_c$ position vector of the point C.
I: moment of inertia around the pendulum's centre of mass
w: angular velocity ($w = \dot{\theta}$)

$$\vec{r}_c = -(x - L\sin\theta)\vec{i} - L\cos\theta\vec{j} \quad \text{(II.23)}$$
$$\vec{v}_c = -(\dot{x} - L\cos\theta\dot{\theta})\vec{i} + Lsen\theta\dot{\theta}\vec{j} \quad \text{(II.24)}$$
$$\vec{v}_c \bullet \vec{v}_c = \left[-(\dot{x} - L\cos\theta\dot{\theta})\vec{i} + Lsen\theta\dot{\theta}\vec{j}\right] \bullet \left[-(\dot{x} - L\cos\theta\dot{\theta})\vec{i} + Lsen\theta\dot{\theta}\vec{j}\right] =$$
$$= \dot{x}^2 - 2\dot{x}L\dot{\theta}\cos\theta + L^2\dot{\theta}^2\cos^2\theta + L^2\dot{\theta}^2 sen^2\theta = \dot{x}^2 - 2\dot{x}L\dot{\theta}\cos\theta + L^2\dot{\theta}^2 \quad \text{(II.25)}$$

So $T_{PENDULUM} = \frac{1}{2} m\left(\dot{x}^2 - 2\dot{x}L\dot{\theta}\cos\theta + L^2\dot{\theta}^2\right) + \frac{1}{2} I\dot{\theta}^2$ (II.26) and finally the total kinetic energy is: $T = T_{CART} + T_{PENDULUM} = \frac{1}{2} M\dot{x}^2 + \frac{1}{2} m\left(\dot{x}^2 - 2\dot{x}L\dot{\theta}\cos\theta + L^2\dot{\theta}^2\right) + \frac{1}{2} I\dot{\theta}^2$ (II.27)

The potential energy of the system is:

$V = -mgL\cos\theta$ (II.28)

Hence the Lagrangian is given by

$$L = T - V = \frac{1}{2} M\dot{x}^2 + \frac{1}{2} m\left(\dot{x}^2 - 2\dot{x}L\dot{\theta}\cos\theta + L^2\dot{\theta}^2\right) + \frac{1}{2} I\dot{\theta} + mgL\cos\theta \quad \text{(II.29)}$$

2.3.4. State equations of the system

Using Lagrange's Equation to find the State equations:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\xi}_i}\right)-\frac{\partial L}{\partial \xi_i}=\Xi_i \rightarrow \begin{cases} \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x}}\right)-\frac{\partial L}{\partial x}=\Xi_x \\ \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right)-\frac{\partial L}{\partial \theta}=\Xi_\theta \end{cases} \quad \text{(II.30)}$$

For x:

$$\left.\begin{aligned} &\frac{\partial L}{\partial \dot{x}}=(M+m)\dot{x}-mL\dot{\theta}\cos\theta \\ &\frac{\partial L}{\partial x}=0 \\ &\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x}}\right)=(M+m)\ddot{x}-mL\cos\theta\ddot{\theta}-mL\dot{\theta}(-sen\theta)\dot{\theta} \end{aligned}\right\} (M+m)\ddot{x}-mL\ddot{\theta}\cos\theta+mL\dot{\theta}^2 sen\theta=f(t)+f_{FRIC\_X} \quad \text{(II.31)}$$

For θ:

$$\left.\begin{aligned} &\frac{\partial L}{\partial \dot{\theta}}=(mL^2+I)\dot{\theta}-mL\dot{x}\cos\theta \\ &\frac{\partial L}{\partial \theta}=-mL\dot{x}\dot{\theta}(-sen\theta)+mgL(-sen\theta) \\ &\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right)=(mL^2+I)\ddot{\theta}-mL\ddot{x}\cos\theta-mL\dot{x}(-sen\theta)\dot{\theta} \end{aligned}\right\} (mL^2+I)\ddot{\theta}-mL\ddot{x}\cos\theta+mgLsen\theta=f_{FRIC\_\theta} \quad \text{(II.32)}$$

Like f_{FRIC_X} and f_{FRIC_θ}, due to the fact that viscous friction depends on $\dot{x}$ and $\dot{\theta}$, respectively, and the State equations are defined using these variables, it is necessary to define two new functions f'_{FRIC_X} and f'_{FRIC_θ} where the viscous friction is pulled out.

Therefore the new functions are:

$$f'_{FRIC\_X}(t)=\begin{cases} f_{STATIC\_X}(t) \leftrightarrow \dot{x}=0 \\ f_{COULOMB\_X}(t) \leftrightarrow \dot{x}\neq 0 \end{cases} \quad \text{(II.33)}$$

$$f'_{FRIC\_\theta}(t)=0 \quad \text{(II.34)}$$

Hence the new State equation system is:

$$(M+m)\ddot{x}-mL\ddot{\theta}\cos\theta+mL\dot{\theta}^2 sen\theta+b_x\dot{x}=f(t)+f'_{FRIC\_X}(t)=f_{TOT}(t) \quad \text{(II.35)}$$

$$(mL^2+I)\ddot{\theta}-mL\ddot{x}\cos\theta+mgLsen\theta+b_\theta\dot{\theta}=0 \quad \text{(II.36)}$$

If $Z = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix}$ (II.37) and $\dot{Z} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\theta} \\ \ddot{\theta} \end{bmatrix}$ (II.38) and $\dot{Z} = A \cdot Z + B \cdot U$ (II.39)

The equations are denoted as follows:

$$(M+m)\dot{x}_2 - mL\dot{x}_4 \cos x_3 + mLx_4{}^2 sen x_3 + b_x x_2 = f_{TOT}(t) \qquad \text{(II.40)}$$

$$(mL^2 + I)\dot{x}_4 - mL\dot{x}_2 \cos x_3 + mgLsen x_3 + b_\theta x_4 = 0 \qquad \text{(II.41)}$$

Working out the value of $\dot{x}_2$ in equation (II.40):

$$\dot{x}_2 = \frac{1}{M+m}\left(mL\dot{x}_4 \cos x_3 - mLx_4{}^2 sen x_3 - b_x x_2 + f_{TOT}(t)\right) \qquad \text{(II.42)}$$

Replacing this value in equation (II.41):

$$(mL^2 + I)\dot{x}_4 - \frac{mL\cos x_3}{M+m}\left(mL\dot{x}_4 \cos x_3 - mLx_4{}^2 sen x_3 - b_x x_2 + f_{TOT}(t)\right) + mgLsen x_3 + b_\theta x_4 = 0 \quad \text{(II.43)}$$

$$(mL^2 + I)\dot{x}_4 - \frac{m^2L^2\cos^2 x_3}{M+m}\dot{x}_4 + \frac{m^2L^2\cos x_3 sen x_3}{M+m}x_4{}^2 + \frac{mL\cos x_3 b_x}{M+m}x_2 - \frac{mL\cos x_3}{M+m}f_{TOT}(t) + \\ + mgLsen x_3 + b_\theta x_4 = 0 \quad \text{(II.44)}$$

$$\frac{(mL^2 + I)(M+m) - m^2L^2\cos^2 x_3}{M+m}\dot{x}_4 = -\frac{m^2L^2\cos x_3 sen x_3}{M+m}x_4{}^2 - \frac{mL\cos x_3 b_x}{M+m}x_2 + \\ + \frac{mL\cos x_3}{M+m}f_{TOT}(t) - mgLsen x_3 - b_\theta x_4 \quad \text{(II.45)}$$

Naming $\boxed{k = (mL^2 + I)(M+m) - m^2L^2\cos^2 x_3}$ (II.46) and working out the value of $\dot{x}_4$:

$$\dot{x}_4 = -\frac{m^2L^2\cos x_3 sen x_3}{k}x_4{}^2 - \frac{mL\cos x_3 b_x}{k}x_2 + \frac{mL\cos x_3}{k}f_{TOT}(t) - \\ - \frac{(M+m)mgLsen x_3}{k} - \frac{(M+m)b_\theta}{k}x_4 \quad \text{(II.47)}$$

Replacing this value in equation (II.42):

$$\dot{x}_2 = \frac{1}{M+m}\left[mL\cos x_3\left(-\frac{m^2L^2\cos x_3 sen x_3}{k}x_4{}^2 - \frac{mL\cos x_3 b_x}{k}x_2 + \frac{mL\cos x_3}{k}f_{TOT}(t) - \right.\right. \\ \left.\left. - \frac{(M+m)mgLsen x_3}{k} - \frac{(M+m)b_\theta}{k}x_4\right) - mLx_4{}^2 sen x_3 - b_x x_2 + f_{TOT}(t)\right] \quad \text{(II.48)}$$

Finally the equations that describe the dynamics of the pendulum have been presented. Therefore, the model of the pendulum block is finished and we can continue modelling next block: Encoders. Although, to be precise, the pendulum block model is not finished yet because these equations are symbolic, but for the moment it is enough. Just after finish with encoders block, we are continuing with finding out all parameters values for these symbols.

2.4. Modelling Encoders

The last block to model is the Encoders block. There are two incremental rotary encoders in this block, one for the position of the cart, x, and other for the angle of the pendulum, θ.

First of all, we are going to explain the incremental rotary encoder behaviour. An incremental rotary encoder converts the angular position of a shaft in a digital code, TTL signals in this case. This angular position or angle is not an absolute position, the encoder does not know where the initial position is, it only knows how the increment of the angular position and it sends this information to other agent who knows the initial position and, consequently, knows the absolute position. This is the reason why before starting a demonstration with the real pendulum is necessary homing the Power Driver Controller using its software.

The homing procedure is very important in the real system, because if the turn-on position of the cart is near to the CKK Compact Module boundaries and nobody cares to the absolute position, the cart could crash with this boundary breaking the CKK Compact Module. Fortunatelly, Power Driver Controller cares to the absolute position of the cart. Note that in the goal software/hardware model of the real system this cannot happen because the initial/turn-on position of the cart will be always zero (in the middle of the CKK Compact Module). Indeed, the incremental encoders can be assumed as absolute in this model due to this fixed initial position of the cart.

Resuming the explanation of an incremental encoder: An incremental encoder always sends two output signals in quadrature (90 degrees out of phase), the reason is to distinguish the direction, rather, the sense of the angular position increment. Please note that this 90 degrees out of phase between both signals, from this point will be called Channel A and Channel B, is not constant, this means that sometimes the Channel A will be 90 degrees in phase ahead of Channel B and other times it will be the opposite (-90 degrees out of phase). The Figure II.11 is shown to illustrate channels A and B with 90 degrees out of phase.

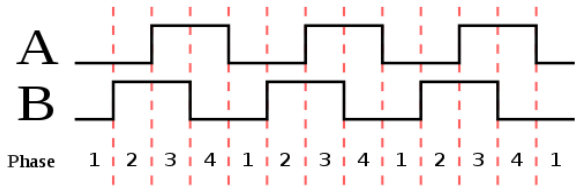


Figure II. 11: Channels A and B of an incremental encoder with 90 degrees out of phase (B ahead A)

Next question to explain is how are generated the TTL output signals of the incremental rotary encoder. In the SCT pendulum system case, both incremental rotary encoder are optical encoders. Therefore, the way to generate the train of pulses is simple: there is a LED which always is emitting light, in front of it there is a photodiode that receives this light converting it in a TTL signal, and between LED and photodiode there is a disk, which is fixed to the shaft, with several bands, half transparent bands and half opaque bands. These bands are

placed alternately, as seen in Figure II.12, with the aim of generate the train pulses when the shaft is moved: when a transparent disk band is between LED and photodiode, the output will be high-level and low-level if the disk band is an opaque one.

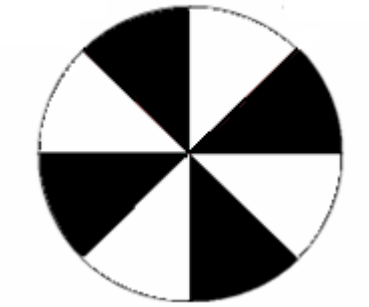


Figure I. 12: Optical encoder disk: transparent (white) and opaque (black) bands

As said previously, the incremental encoder has two outputs in quadrature, Channel A and B, and depending on the angle sense one channel leads the other. Inevitably, two sets of LED/photodiode are necessary, which could be a problem for constructing encoders: 90 degrees out of phase are needed, this means a physic distance between sets of half band, if the resolution is high and, consequently, the number of bands is high too, the encoder is technically impossible to build. Only as reminder, resolution is the minimum change at the input that is possible to measure at the output.

Apparently, incremental encoders are worse than absolute, however, quadrature outputs can be used to increment four times the encoder resolution (counting the number of rising and falling edges in both channels, instead the number of pulses) and using a different type of encoder disk, see the Figure II.13, is possible to obtain higher resolution than the absolute encoders.

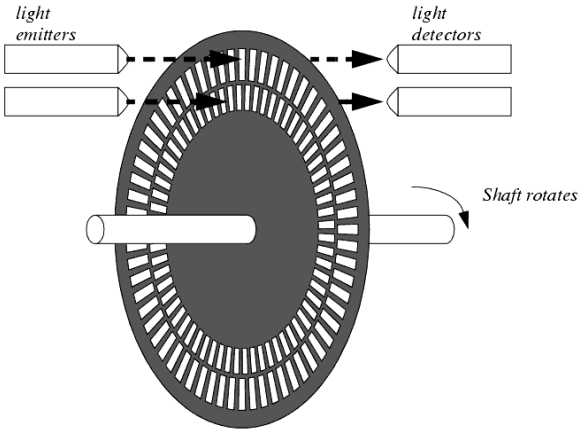


Figure II. 13: Encoder disk of an incremental rotary encoder

In this point, we are able to model the encoders block. The first step is to explain the simplifications that were done in these encoders model:

- The encoder disk used is a simplification of the real encoder disk (shown in the previous figure).
- The sets of LED/photodiode are spaced half band.

- Although in the encoder disk there are many bands, exploiting the symmetry of the disk, only two have been considered as work/operation area for the encoder model. An explanatory diagram is shown in Figure II.14.

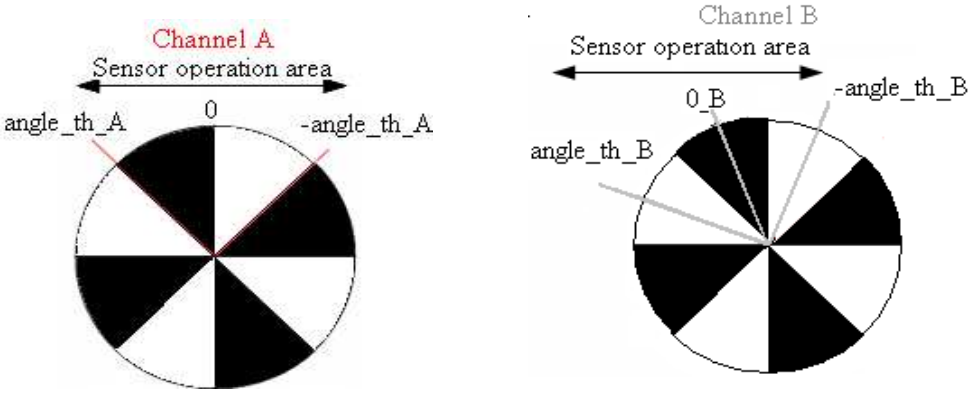


Figure II. 14: Sensor operation areas for both channels in our encoder model

The main idea of this encoder model is that if the increment angular position of the shaft exceeds the boundaries shown in the diagram, then this angular position will be corrected to return to be within the operation area. This correction consists on add/subtract an angle equivalent to two bands depending on the angular position has exceeded the left/right boundary.

Next step is to obtain the value of angle_th_A, or the value of disk band in radians, for which is necessary to remember the resolution values of both encoders: 500 lines/rev for *x* and 5000 lines/rev for θ. As each line is equivalent to an opaque line, please remember the resolution definition, there are 1000 bands/rev into the *x* encoder disk and 10000 bands/rev into the θ encoder disk.

$$angle\_th\_A\_x = \frac{2\pi\left[rad/rev\right]}{1000\left[bands/rev\right]} = 2\pi\left[mrad\right] \quad \text{(II.49)}$$

$$angle\_th\_A\_\theta = \frac{2\pi\left[rad/rev\right]}{10000\left[bands/rev\right]} = 0.2\cdot\pi\left[mrad\right] \quad \text{(II.50)}$$

These constant values are included in our Matlab script of model constants that is preloaded just before start the model simulation.

Note that for Channel B the values in both encoders are the same of Channel A plus an offset of $0.5\cdot angle\_th\_A$.

Next step is to explain the cart position encoder, because using a rotary encoder for the angle of the pendulum, an angular position, it is normal, however, using a rotary encoder for the position of the cart, a linear position, it is a little bit strange. The question is: how is possible to find out the value of x from the value of φ, the angular position of the Spindle? The answer could be found thinking in infinitesimal calculus.

$$\omega = \frac{\Delta\varphi}{\Delta t}\left[rad/s\right] \text{(II.51)} \qquad v = \frac{\Delta x}{\Delta t}\left[m/s\right] \text{ (II.52)}$$

$$\omega = \frac{2\pi\left[rad/rev\right]}{k\left[m/rev\right]} v\left[m/s\right] = \frac{2\pi}{k}\left[rad/m\right]\frac{\Delta x}{\Delta t}\left[m/s\right] = \frac{\Delta\varphi}{\Delta t}\left[rad/s\right] \text{ (II.53)}$$

$$\frac{2\pi}{k}\cdot\frac{\Delta x}{\Delta t} = \frac{\Delta\varphi}{\Delta t} \rightarrow \Delta x = \frac{k}{2\pi}\cdot\Delta\varphi = \frac{5\cdot 10^{-3}}{2\pi}\cdot\Delta\varphi\left[m\right] \text{ (II.54)}$$

But this answer is from the point of view of the real encoder, in the model case, the situation is inverted, the position of the cart is known because is an output of the pendulum block, and the angle of the Spindle is unknown. Modifying just a little bit the last equation this angular position is found out:

$$\frac{2\pi}{k}\cdot\frac{\Delta x}{\Delta t} = \frac{\Delta\varphi}{\Delta t} \rightarrow \Delta\varphi = \frac{2\pi}{k}\cdot\Delta x = \frac{2\pi}{5\cdot 10^{-3}}\cdot\Delta x\left[rad\right] \text{ (II.55)}$$

Now it is possible to obtain the increments of the pendulum angle and the Spindle angle from the inputs of the encoders block, x and θ, and using this increments it is possible to generate both trains of pulses as outputs.

The only thing that could be a problem is if the increments are bigger than two bands, because this model always add or subtract this quantity when the boundaries have been exceeded. This problem was solved fixing the sample period sufficiently small to avoid that these increments exceed $2\cdot angle\_th\_A$.

To calculate these sample periods it is necessary to know the maximum angular velocities of the pendulum and Spindle. For finding out the maximum angular velocity of the pendulum an experiment was done: the pendulum was hard pushed and the number of revolutions in one second was recorded. Four revolutions per second was the maximum angular velocity of the pendulum, we want to call the reader attention to realise that when the real pendulum falls from the balanced position never exceed this maximum angular velocity.

Therefore:

$$\omega_{\theta\_MAX} = 4[rev/s]\cdot 2\pi[rad/rev] = 8\pi = 25.13274[rad/s] \text{ (II.56)}$$

$$\omega_{\theta\_MAX} = \frac{\Delta\theta_{LIMIT}}{\Delta t_{\theta\_MAX}} \rightarrow \Delta t_{\theta\_MAX} = \frac{\Delta\theta_{LIMIT}}{\omega_{\theta\_MAX}} = \frac{2\cdot angle\_th\_A\_\theta}{\omega_{\theta\_MAX}} = \frac{2\cdot 0.2\cdot\pi\cdot 10^{-3}}{8\pi} = 5\cdot 10^{-5}[s] \text{ (II.57)}$$

For the maximum angular velocity of Spindle, the known value is the maximum linear velocity supported by the CKK Compact Module, so the equations are:

$$\omega_{\varphi\_MAX} = \frac{2\pi}{k}\cdot v_{\_MAX} = \frac{2\pi[rad/rev]}{5\cdot 10^{-3}[m/rev]}\cdot 0.55[m/s] = 691.15[rad/s] \text{(I.58)}$$

$$\omega_{\varphi\_MAX} = \frac{\Delta\varphi_{LIMIT}}{\Delta t_{\varphi\_MAX}} \rightarrow \Delta t_{\varphi\_MAX} = \frac{\Delta\varphi_{LIMIT}}{\omega_{\varphi\_MAX}} = \frac{2\cdot angle\_th\_A\_x}{\omega_{\varphi\_MAX}} = \frac{2\cdot 2\pi\cdot 10^{-3}}{691.15} = 1.82\cdot 10^{-5}[s] \text{(I.59)}$$

After these calculations the maximum sample period, which should be at most the smallest value of them, is fixed to: $Ts = 10\mu s$. It is equivalent to say that the minimum frequency of the software model is fixed to: $f = 100KHz$.

Such as we can fix the model frequency to any value lower than 33MHz (maximum frequency of Xilinx Virtex 2 Pro FPGA), we have decided to fix the encoder model frequency to 1MHz ($Ts = 1\mu s$) because in this way we have a margin of security. If it is possible always it is better to design with a margin because being close to the limits sometimes cause avoidable errors.

To finalize the modelled of this encoders block, it is necessary to explain how will be generated the outputs and also how will be managed the out of boundaries situations. Both answers are the same: Finite State Machine (FSM). It will be a Moore FSM, this means that their outputs depend on the current state. This FSM will have as inputs: the angular position and current state; and as outputs: the train pulses and next state.

Four states are necessary: Lightness, transparent band; Darkness, opaque band; Over_upper_limit, the angular position needs to be adjusted subtracting 2*$angle\_th$; and Lower_lower_limit, the angular position needs to be adjusted adding 2*$angle\_th$.

Choosing Moore or Mealy for the FSM is unimportant for us. But Moore criteria was selected because it is simpler than Mealy, remember that in Moore machine the outputs depend only on the current state and in Mealy depend also in inputs values. The main idea is checking the angular position to decide the next state (Lightness, Darkness, Over_upper_limit or Lower_lower_limit) each sample period for avoiding the increment of angle exceeds $2 \cdot angle\_th$.

Matlab StateFlow Tool has been used to develop the four FSM that are included in our software encoder model (each encoder model has two FSM, one for each channel). This Toolbox is very intuitive as is shown in Figure II.15 and II.16.

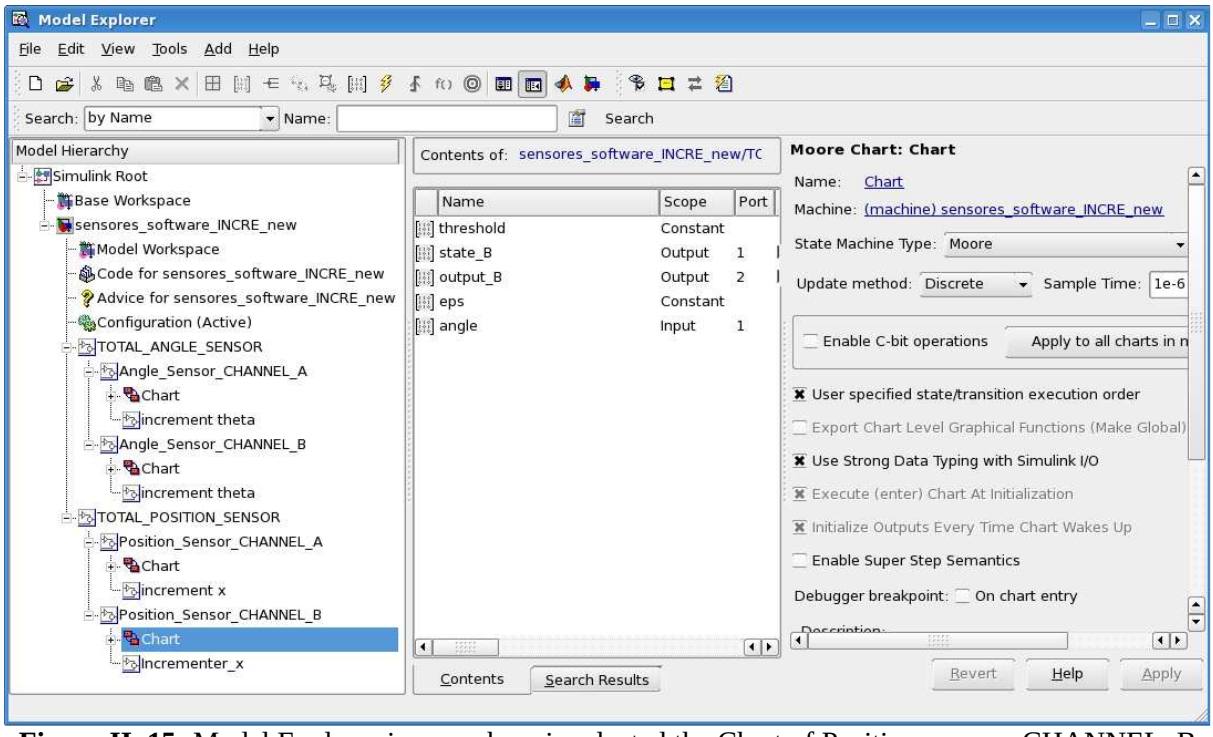


Figure II. 15: Model Explorer image where is selected the Chart of Position_sensor_CHANNEL_B

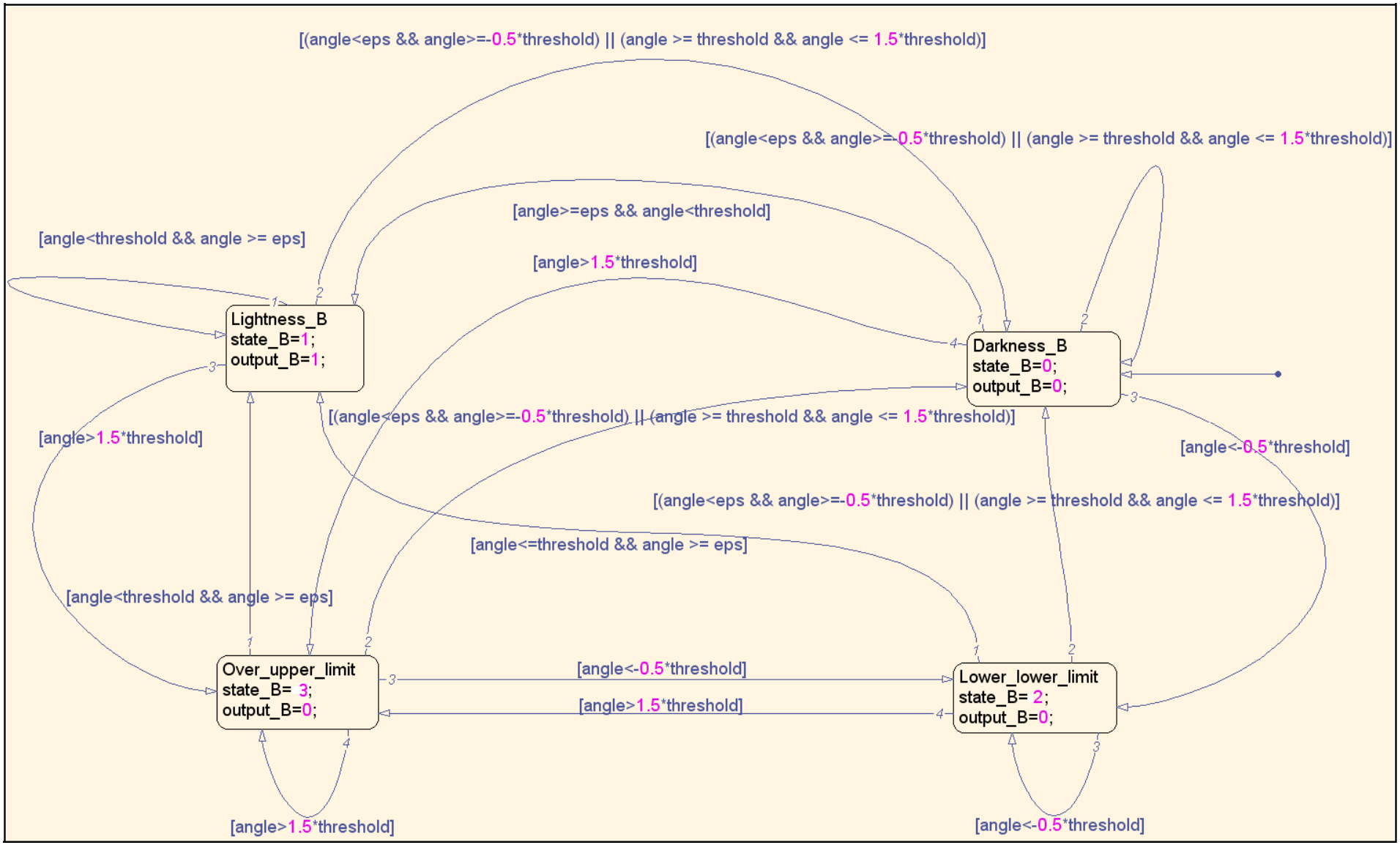


Figure II. 16: StateFlow diagram Channel B for Cart position or Angular position encoder (depending on threshold value)

In Figure II.15 looking from left to right we can see the four FSM Charts at Model Hierarchy, in this case is selected the Chart of Position_sensor_CHANNEL_B, in the middle the contents of this Chart are shown: threshold, which is defined as constant and its value depends on the type of encoder (II.49 for cart position or II.50 for angular position); state_B, which is an output that indicates state of the FSM for the next sample period and it is necessary for angle readjustment; output_B, which is set to 'high' or 'low' level depending on the state and creating the train pulses output; eps, which is a very small constant that is used as replacement for zero value because it gives some problems; and finally angle, which is the FSM input inherited from Simulink and depending on it is set the next state. At right side in Figure II.15 you can select State Machine Type (Moore/Mealy), Update method (Discrete/Continuous/Inherit) and Sample Time (only for Discrete Update method).

Figure II.16 shows the StateFlow of the FSM of Moore for the channel B in the cart position encoder using the Matlab StateFlow Tool. Here the four possible states for the FSM appear like rectangular boxes, which include the name of the state and its two associated outputs. The arrows indicate the transitions between states depending on the angle input value. This is an example for channel B, the StateFlow Chart for channel A is the same but replacing the transition conditions: *-0.5* threshold* by *–threshold* and *1.5* threshold* by *threshold*. Recalling that the only differences between cart position encoder and angular position encoder concerning FSM charts is the value of the constant *threshold*, I.49 for the first one and I.50 for the second one.

The last point to explain in this incremental rotary encoder model is the extra blocks that complement the FSM block to obtain a good model. It is necessary an incrementer block that obtains the increment of angle in the encoder disk. It is immediate for the angular position (θ) but for the cart position (x) encoder the incrementer block gives the increment of x, to obtain the increment of angle it is necessary a gain block, remember Equation II.55. The angle that is used as input of the FSM block will be obtain as the addition of the last angle value plus the increment of angle ($\Delta\varphi$). See Equation II.60.

$$\varphi_{CURRENT} = \varphi_{LAST} + \Delta\varphi[rad] \quad \text{(II.60)}$$

This way to obtain the value of angle requires memory blocks. And finally, it is necessary an angle readjustment block for keeping this value inside the operation area of the encoder (see Figure II.14). Remember that this area is equal to two bands of the encoder disk, when the angle is out of these boundaries an readjustment is necessary: adding 2**angle_th* when the angle is lower than the lower limit (state Lower_lower_limit) or subtracting 2**angle_th* when the angle is higher than the upper limit (state Over_upper_limit). Therefore, this readjustment block needs as inputs: the current state and angle; because it is necessary to know when and what we apply the readjustment.

After the explanation of the incrementer block for the angle of the encoder disk, all important ideas for modelling the incremental rotary encoders have been explained, although, in the third chapter of this master thesis these ideas will be further analysed.

At this point, it seems that we can end the chapter 2. Modelling the Pendulum, but as we stated in Section 2.3. Modelling the Pendulum by using differential equations, it is necessary to find numerical values of the parameters used in the symbolic equations that describe the pendulum. And, thereby, this is the next step: Characterization of SCT pendulum.

2.5. Characterization of SCT pendulum: Non linear Grey Box Model. Calculation and identification of parameters

Once this section has been reached, we could say that the equations that describe the pendulum have been found, but at the moment they are symbolic equations. The question is: do we know the value of all these symbols? For sure, some of them are known or are really easy to find out, for instance the gravity acceleration *g* or the mass *m* (mass of bob plus rod). Nevertheless, there are other parameters that are unknown or impossible to measure using SCT laboratory resources, such as the viscous coefficient in θ, b_θ.

There are three different ways for modelling dynamic systems: white box models, which are based on first principles, for instance, dynamic systems that are described by Newton equations, where all parameter values are known; black box models, where there are not prior information available (equation or parameters values are unknown) and it is necessary system identification from measurements of the dynamic systems to find out the model; grey box models, which are based on some prior information but this information is not complete and it is necessary system identification for finding out the parameter values which are unknown.

In this case, it is a grey box model, which usually represents a physical model (the pendulum equations previously obtained) with some known parts (white parameters) and other unknown ones (black parameters) which need to be approximated. In this case, the model is a non linear grey box model because the equations are non linear. Please note that it is not possible to linearize the model because it is necessary an accurate model that works in whole space state, this means big angle θ values, then it is not possible to linearize sinus and cosines using small angles or Taylor approximations.

In this work a non linear grey box model was chosen but there is a different way to model the SCT pendulum: subdividing the space state in multiple small angles (for example, one degree) and applying linearizations to find out several linear grey box models (360 models in this case). The price to pay using this different solution is: to switch between models depending on the angle. This is the main reason why this different solution was not chosen.

Next step is to enumerate and classify all the model parameters into known (type K) and unknown (type U), more specifically, into parameters that could be calculated/measured and parameters that could not and need to be identified.

Parameter	Meaning	Unit	Type	Value
g	Gravity acceleration	m/s^2	K	9.80665
L	Distance from pivot to the centre of mass of rod	m	K	-
I	Moment of inertia of the pendulum	$Kg \cdot m^2$	K	-
b_x	Viscous coefficient of the cart	$N \cdot s/m$	K	-
b_θ	Viscous coefficient of the pendulum	$N \cdot s/rad$	U	-
M	"Virtual" mass of the cart (including moments of inertia)	Kg	K	103.205
m	Mass of the pendulum (rod + bob)	Kg	K	0.581
μ_C	Coulomb friction coefficient	-	K	-
μ_S	Static friction coefficient	-	K	-

α	Voltage to force conversion factor	N/V	K	51.525
v_S	Stribeck velocity*	m/s	U	-
γ	Form factor*	-	U	-
V_{off}	Offset voltage	V	K	5

* Exponential friction model will be explained later

Table 1: Parameters of pendulum model before characterization

At this point, the next step is to calculate the value of known model parameters:

2.5.1. CALCULATION OF CENTRE OF MASS

First of all, the origin of coordinate in our system has to be determined. It is situated in the pivot of the pendulum, which is inserted in the back side of Piece 1 (not shown in Figure II.2). The origin of coordinates is placed at the back side of the piece as shown in Figure II.2. The diagram below, Figure II.17, shows the representation of the said back side:

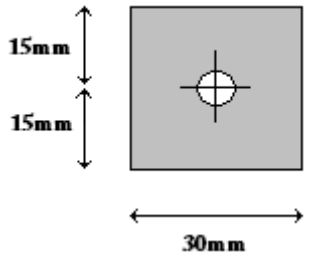


Figure II. 17: Back side of Piece 1 shown in Figure II.2

Secondly, the rod (Piece 2 in Figure II.2) is inserted in Piece 1 15mm deep and 40mm in Piece 3. Therefore the final length of the rod, L_{2_F}, is 186.5mm.

The distance to the centre of mass is given by the next formula:

$$L_{CM} = \frac{\sum_{i=1}^{N} \tilde{y}_i \cdot m_i}{\sum_{i=1}^{N} m_i} \text{ With i = 1, …, N. (II.61)}$$

Naming N as the number of pieces, m_i each mass and $\tilde{y}_i$ the distance from the centre of mass of each piece to the origin of coordinates.

$$\tilde{y}_1 = 0 \text{ (II.62)} \quad \tilde{y}_2 = \frac{241.5}{2} = 120.75mm \text{ (II.63) and } \tilde{y}_3 = 241.5 - \frac{40}{2} = 221.5mm \text{ (II.64)}$$

$$L_{CM} = \frac{0 \cdot 185.155 + 120.75 \cdot 69.457 + 221.5 \cdot 326.388}{185.155 + 69.457 + 326.388} = 138.867mm = 13{,}88cm \text{ (II.65)}$$

2.5.2. CALCULATION OF MOMENT OF INERTIA

The next parameter value to be found is *I*, that is, the moment of inertia in the centre of mass. In the beginning an approximated calculation has been done, only to know its order of magnitude. Consequently, the mass of the rod, m_2, have been neglected:

$$I_{approx} = m_1 \cdot d_1^{\,2} + m_3 \cdot d_3^{\,2} \text{ (II.66)}$$

The distances from the centre of mass of each piece to the total centre of mass are:

$$d_1 = L_{CM} = 0.1388672m \text{ (II.67)}$$
$$d_3 = L_2 - L_{CM} - 0.02 = 0.2415 - 0.1388672 - 0.02 = 0.082633m \text{ (II.68)}$$

So finally:

$$I_{approx} = m_1 \cdot d_1^{\,2} + m_3 \cdot d_3^{\,2} = 0.005799Kgm^2 \text{ (II.69)}$$

This value is only an approximate value. In order to obtain the exact value of this moment of inertia it is necessary to apply the Steiner Theorem[15]. Remember that this theorem is used to find out the moment of inertia of an object about any parallel axis through its centre of mass. Calling I_{O_i} the moment of inertia of the element *i* in its centre of mass, and I_{D_i} the moment of inertia of the element *i* that is generated due to the distance from its centre of mass to the total centre of mass of the system, the following formulas are obtained:

$$I_{TOT\_i} = I_{O\_i} + I_{D\_i} \text{ (II.70)}$$
$$I_{TOT} = \sum_{i=1}^{N} I_{TOT\_i} \text{ (II.71)}$$

Next step is to calculate the distances between the centre of mass *i* and the total centre of mass:

$$d_1 = L_{CM} = 0.01388672m \text{ (II.72)}$$
$$d_2 = L_{CM} - \left(\frac{L_{2\_F}}{2} + 0.015 \right) = 0.01388672 - \left(\frac{0.1865}{2} + 0.015 \right) = 0.030617m \text{ (II.73)}$$
$$d_3 = L_{2\_F} + 0.015 + 0.02 - L_{CM} = 0.1865 + 0.015 + 0.02 - 0.01388672 = 0.082633m \text{ (II.74)}$$

$$I_{D\_i} = m_i \cdot d_i^{\,2} (I.75) \rightarrow \begin{cases} I_{D\_1} = 0.003571Kgm^2 (I.76) \\ I_{D\_2} = 0.000065Kgm^2 (I.77) \\ I_{D\_3} = 0.002229Kgm^2 (I.78) \end{cases}$$

To calculate the moment of inertia of each element, the well-known equations for a rod (*i* = 2), a cylinder (*i* = 3) and a parallelepiped (*i* = 1) are used:

Rod:

$$I_C = \int_{-L/2}^{L/2} \frac{M}{L} x^2 dx = \frac{1}{12} M \cdot L^2 \text{ (II.79)}$$

In this case: $I_{O\_2} = \frac{1}{12} m_2 \cdot L_{2\_F}^2 = 0.000201 Kgm^2$ (II.80)

Cylinder:

$$I_C = \int_{-L/2}^{L/2} \left(\frac{1}{4} R^2 + x^2 \right) \frac{M}{L} dx = \frac{1}{4} M \cdot R^2 + \frac{1}{12} M \cdot L^2 \text{ (II.81)}$$

Deriving a specific case from the general: $I_{O\_3} = \frac{1}{4} m_3 \cdot R_3^2 + \frac{1}{12} m_3 \cdot L_3^2 = 0.000117 Kgm^2$ (II.82) with $R_3 = 0.03m$ (II.84) and $L_3 = 0.04m$ (II.85)

Parallelepiped:

$$\int_{-c/2}^{c/2} \left(\frac{1}{12} b^2 + x^2 \right) \frac{M}{c} dx = \frac{M}{12} \left(b^2 + c^2 \right) \text{ (II.86)}$$

In this specific case, the parallelepiped is a cube so: $I_{O\_1} = \frac{1}{12} m_3 \cdot 2L_1^2 = 0.000028 Kgm^2$ (II.87) with $L_1 = 0.03m$ (II.88)

Finally, the value of the moment of inertia of the pendulum has been found out:

$$\boxed{I = I_{TOT} = \sum_{i=1}^{N} I_{O\_i} + I_{D\_i} = 0.00621034 Kgm^2} \text{ (II.89)}$$

2.5.3. CALCULATION OF FRICTION COEFFICIENTS

As was said previously, the friction forces are usually neglected but, as was demonstrated in [14], to neglect the friction introduces differences with reality (remember the ripple observed in the Figure II.10). For sure, these inaccuracies are not desirables for us because our objective is to find out a realistic and accurate model of the SCT pendulum. For this reason, the calculation of these friction coefficient values is important in our pendulum model and it makes this master thesis a bit different from most of consulted literature (e.g.: [9] [10] [11] [12] [13]), where these coefficients have been neglected.

To determinate the values of these coefficients there are several methods, for instance checking coefficients tables, in this case steel-steel, or finding them by doing an experiment. This last option was the chosen one because it will be more specific and accurate for our model than the others. Two different experiments were done in order to find out the values of μ_s, μ_c and b_x coefficients.

STATIC COEFFICIENT

First of all, for this experiment the pendulum structure from the cart has been detached because the static friction coefficient does not depend on the mass, and detaching it is much simpler. The last step before the experiment can begin is connecting a variable voltage source as the input of the Power Driver.

This experiment consists in applying an incremental voltage that starts with 5 volts because this is the offset voltage, and observing when the cart starts to move (an input voltage higher than 5V will move the cart to the left, and to the right if the input voltage is less than 5V).

After recording this voltage value it is possible to convert it into an input force using α constant, and obtaining the value of the coefficient from the following equation $\left|F_{StaticFriction}(t_0)\right| = \left|F_{Input}(t_0)\right|$ (II.90) where t_0 is just the moment when the cart starts to move.

$$\left|\mu_s \cdot M \cdot g\right| = \left|\alpha \cdot \left(V_{in} - V_{Offset}\right)\right| \text{(II.91)}$$

Such as the static friction coefficient could depend on sense of movement this experiment has been carried out two times: moving the cart to the left (V^+_{in}) and to the right (V^-_{in}). Therefore, we will obtain two different values for this coefficient. Unfortunately, in our model we have considered only one value for this coefficient (independently of sense of cart movement) then the best solution is to calculate the mean of these values.

The input voltages, measured after the ADC, were $V^+_{in} = 5{,}705V$ (II.92) and $V^-_{in} = 4{,}248V$ (II.93). Therefore, the coefficient values and their mean are determined:

$$\mu^+_s = \frac{1}{M \cdot g}\alpha \cdot \left|V^+_{in} - V_{Offset}\right| = 0.03589 \text{ (II.94)}$$

$$\mu^-_s = \frac{1}{M \cdot g}\alpha \cdot \left|V^-_{in} - V_{Offset}\right| = 0.03828 \text{ (II.95)}$$

$$\boxed{\mu_s = \frac{\mu^+_s + \mu^-_s}{2} = 0.03709} \text{ (II.96)}$$

DINAMIC FRICTION AND VISCOUS COEFFICIENT

The next experiment is designed to find out the values of the μ_c and b_x parameters. The preparations for the experiment are the same as in the case of the static friction coefficient. But in this case, we have two targets: the first one is to find out the cart velocity; and the second one, in order but not in importance because it is the main target, is to find out the dynamic and viscous coefficient using the cart velocity value.

The reason why it is necessary to determine the velocity in advance is in the equilibrium equation:
$-F_{Friction}(t_0) = F_{Input}(t_0)$ (II.97) Where t_0 is just the moment when the cart is moving with constant velocity.

If we define the friction force as in the modelling the pendulum section, the Equation II.98 is obtained:

$$\mu_c \cdot M \cdot g + b_x \cdot \dot{x} = \alpha \left| V_{in} - V_{off} \right| \quad \text{(II.98)}$$

As seen in Equation II.98 if the velocity ($\dot{x}$) is known there are only two unknown quantities, μ_c and b_x, and it is possible to solve this equation as long as we add another new equation, more exactly, the same equation but having different values of input voltage and velocity.

Thereby, the first step in our experiment is to set a constant voltage as input and secondly the cart position data generated by the sensors will be collected (see Figure II.18).

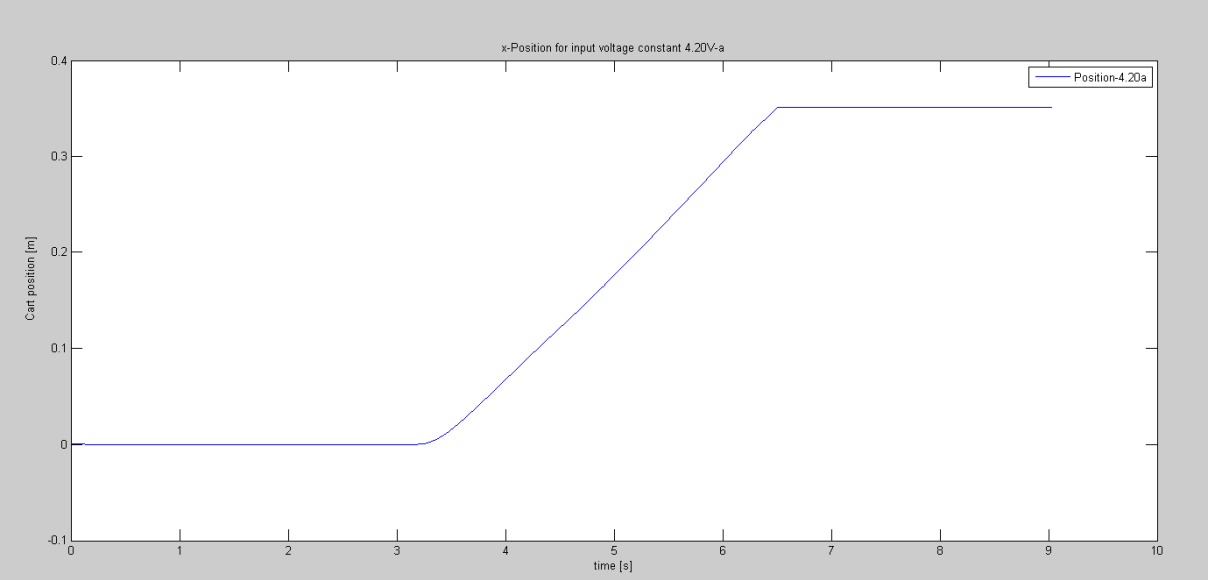


Figure II. 18: Cart position Vs time graph for a constant input voltage of 4.20V

After collecting these data, we will use a Matlab script that basically works by loading the position data, selecting their interesting points: where the position starts to increase/decrease from zero following a straight line (in Figure II.18 from 3.2 to 6.5 seconds); and finding out the target velocity.

To find this velocity the Matlab function “polyfit” is used. This is a polynomial curve fitting function in Matlab that, after receiving the data, finds the best coefficients of a polynomial of degree n in least square sense. Remember that in a position straight line Vs time graph, the slope of this function corresponding to the velocity.

Such as we need to fit a straight line that corresponds to position data, it involves using “polyfit” with the argument called number of degree equal one (fitting through a linear polynomial).

As we said, this function returns the best fitting curve for our data straight line giving the polynomial coefficients as a vector. In linear polynomial case, this returned vector has a size of two: first one is the coefficient corresponding to the “*n*” term and the second one is the coefficient corresponding to to the “*n* - 1” term (remember n = 1).From these returning values we will be only interested in the first one because it corresponds to the slope of the position data line, which is the target velocity of this experiment.

Only to show graphically how works the Matlab "polyfit" function and also how is determined the velocity in our Matlab script, we have used Matlab Curve Fitting Toolbox, as Figure II.19 shows.

After selecting the interesting points, in this case as is shown in Figure II.19, from 3.5 seconds, when the cart begins to move, until 6.5 seconds; using linear polynomial (this is equivalent to "polyfit" with n = 1) a red straight line is drawn, which is the fit linear polynomial, and the results are obtained, we can see them in the middle of left figure in Figure II.19. The first coefficient (called p1 in Figure II.19) corresponds with the slope of the fit straight line and it is the target velocity given in meters per second (0.1132 in this case).

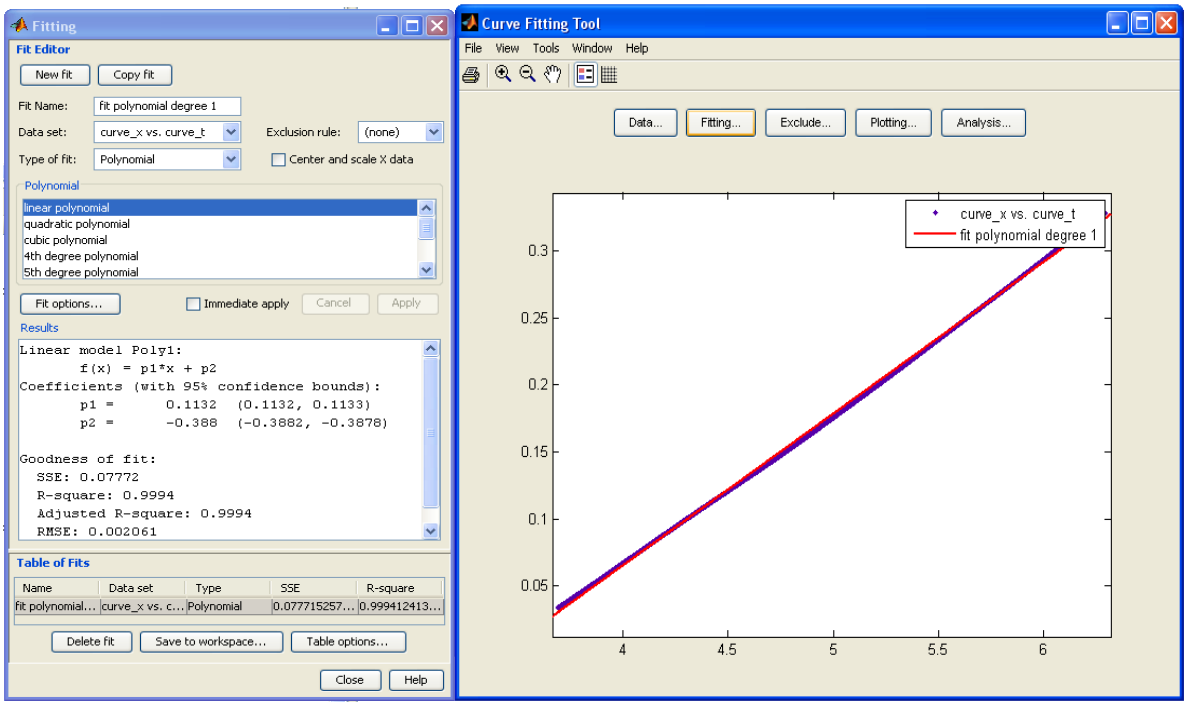


Figure II. 19: Matlab Curve Fitting Toolbox figures used to find out the cart velocity

As there are some random changes from one experiment to other, six different input voltages were taken, half of them higher than 5V and the other half lower. Each value is separated from the offset voltage in absolute value, from 0.8 to 1.8V, jumping with a step of 0.2V. Besides each input voltage value was experimentally repeated three times, denoted as a, b and c cases. The results are shown in Table 2, where there are five columns: input voltage, velocity in the experiment a, in b, in c and the mean of these three velocities.

Vin [V]	Velo_a [m/s]	Velo_b [m/s]	Velo_c [m/s]	Velo_mean [m/s]
4.2	0.1132	0.1105	0.1135	0.1119
6	0.1669	0.1682	0.1745	0.1699
3.8	0.2492	0.2506	0.2567	0.2522
6.4	0.3233	0.3248	0.3256	0.3246
3.4	0.4083	0.4136	0.4201	0.4140
6.8	0.4728	0.4743	0.4799	0.4757

Table 2: Velocities obtained experimentally for finding out Coulomb and Viscous friction coefficient

It is important to be sure that the maximum velocity reached in this experiment is below the permissible speed given in the datasheet of the Compact Module [6]. The Figure II.20 shows this value (see the line for 12x5):

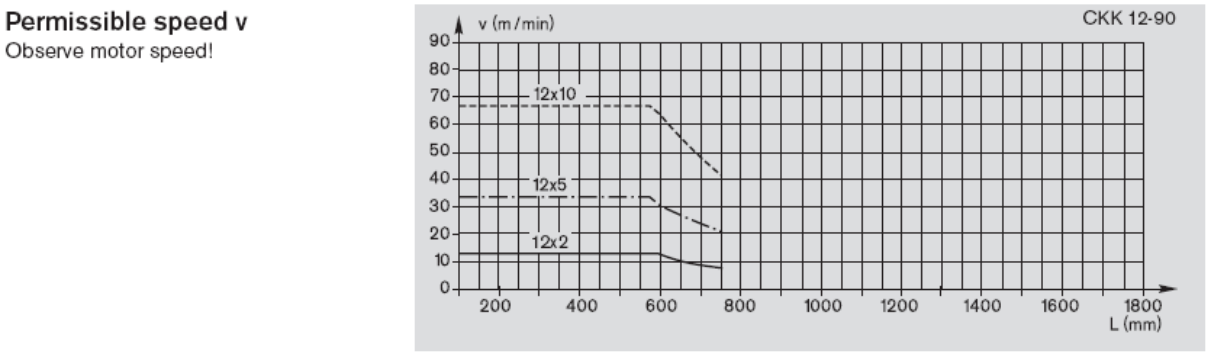


Figure II. 20: Graph from CKK 12-90 datasheet of permissible velocity at the Compact Module

If we get the maximum velocity value from the table and convert it into the same units that the permissible speed has, we will be able to check the condition mentioned before:

$$\text{Velo\_max} = 0.4757\ [\text{m/s}]\ 60[\text{s/min}] = 28.54 < 33\ [\text{m/min}] \quad \text{(II.99)}$$

Now, with these mean velocities it is possible to calculate the values of the dynamic friction coefficients, also called Coulomb coefficient (μ_c) and the cart viscous friction coefficient (b_x). A Matlab script was developed to obtain these values through solving every possible combination of two equations and finally calculating their mean.

The results obtained are:

mean_mu_c = 0.0292 (II.100) and *mean_b_x* = 155.1574 (II.101)

Therefore, the table of parameters is updated:

Parameter	Meaning	Unit	Type	Value
g	Gravity acceleration	m/s^2	K	9.80665
L	Distance from pivot to the centre of mass of rod	m	K	0.138867
I	Moment of inertia of the pendulum	$Kg \cdot m^2$	K	6.21034e-3
b_x	Viscous coefficient of the cart	$N \cdot s/m$	K	155.1574
b_θ	Viscous coefficient of the pendulum	$N \cdot s/rad$	U	0.01
M	"Virtual" mass of the cart (including moments of inertia)	Kg	K	103.205
m	Mass of the pendulum (rod + bob)	Kg	K	0.581
μ_C	Coulomb friction coefficient	-	K	0.0292
μ_S	Static friction coefficient	-	K	0.03709
α	Voltage to force conversion factor	N/V	K	51.525
v_S	Stribeck velocity	m/s	U	-
γ	Form factor	-	U	-
V_{off}	Offset voltage	V	K	5

Table 3: Parameters of pendulum model updated

Please note that b_θ has now a necessary value, because the identification method needs a starting point, also called initial value, for b_θ. This value was obtained from the [12]. It is necessary that the starting point is more or less in the same order of magnitude as the real value of this parameter. At this point it is possible to start the identification of the parameter b_θ, which is the only unknown one (note v_S and γ will be used and explained in the future).

2.5.4. PARAMETERS IDENTIFICATION

First of all, parameter identification has been done using System Identification Toolbox of Matlab R2008b. The main idea is to generate a m-file that contains the non linear grey box model, where all the parameters are specified, initialized and selected as *free* (able to identify) or *fixed* (not able to identify). This m-file will create an idnlgrey object, which is a non linear grey box Matlab object for our model, based on a different m-file where the dynamic pendulum equations are described as non linear State Equations (cf. section 2.3.4). For better understanding please look at the Figure II.21.

In order to begin the identification training data are necessary, which are taken from the real pendulum and used to fit the outputs of the model (idnlgrey object) to the real outputs. These training data are obtained from several experiments with the real pendulum, and through an m-file that is used to process data and generate an iddata object, which is a data Matlab object that contains the input data ($V_{IN}(t)$) and the two output data ($x(t)$ and $\theta(t)$) obtained in these experiments. Now it is possible to start the parameters identification.

The Matlab System Identification Toolbox has a graphical interface, but, unfortunately, it does not work for the non linear grey box model, which is our case. There is a Matlab function called "*pem*", standing for iterative prediction-error minimization method, which can work with the non linear grey box models. This "*pem*" Matlab function is the identification method used in this work, which has the idnlgrey_object and the iddata_object as inputs and returns an idnlgrey_object as outputs having different values in the free parameters set on the input, because they have been modified to fit the model and the real system outputs better. This explanation can be a bit difficult to understand. That is the reason why an explanatory flowchart is presented in Figure II.21.

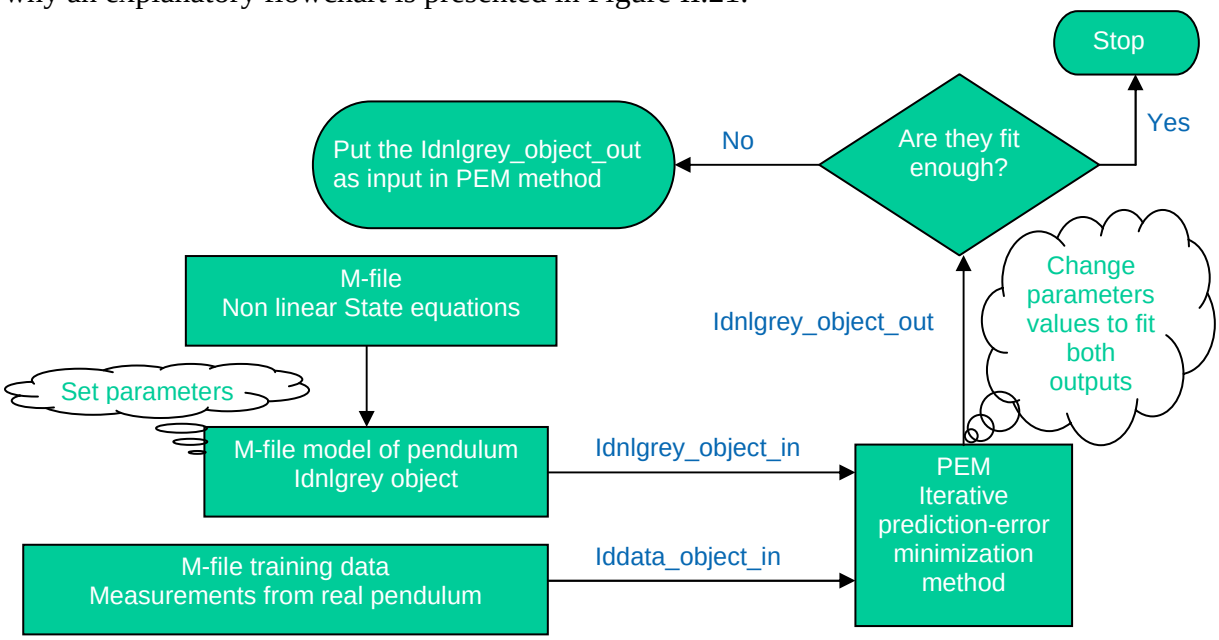


Figure II. 21: Flowchart of the model parameter identification method used

As it is shown the m-file which contains the non linear State equation, which is called "*inv_pendulum_m.m*", is included in the m-file that generates the model object. This file receives as input arguments: *t*, time; *z*, State vector composed of *x*, *x_dot*, *theta*, *theta_dot*; V_{IN}, input voltage; and all the parameters, *g*, *I*, *L*… It gives as output: *z_dot*, State derivative vector; and *y*, vector composed of *x* and *theta* that are the model outputs. The ODE (Ordinary Differential/Difference Equation) solver of Matlab uses the output *z_dot* to give to the file its next input *z*, this is the way to obtain at every time *t* the model outputs, *x* and *theta*, which are inside the vector y. These model outputs will be compared with the real output measured, which are included into the iddata_object.

The m-file that generates the idnlgrey_object, which is called "*creation_idnlgrey_object.m*", executes the constructor method with the following input arguments: **FileName**, the file that describes the model, "*inv_pendulum_m.m*" in this case; **Order**, a vector with the model order, and consists of: the number of outputs, two (*x* and *theta*); the number of inputs, one (V_{IN}); and the number of states, four (*x*, *x_dot*, *theta*, *theta_dot*); **Parameters**, a vector with the initial values of them and the reason why all previous calculations were done; **InitialStates**, a vector with the initial values of the model States, four zeros; and **Ts**, the sample period of the model, the value in this case is zero because the model is a time-continuous system.

After constructing the object, this m-file sets several properties for the model, some of them like InputName, OutputName, Input/OutputUnit, TimeUnit, names and units of parameters are superfluous; however, there is an very important property that is set in this m-file and this one determines the parameters as *free* (able to identify) or as *fixed* (not able to identify). At the moment, all of them, except but b_θ are fixed.

There are also other important model properties that can be set on this file or can be changed its values after constructing the object. They are related to the options for the Estimation Algorithm. They are really important because the quality of the results can be affected depending on these options. Now, the most interesting values for these ODE solver properties for non linear grey box models are shown and explained:

SIMULATION METHOD

Solver - The ODE solver used to solve the state space equations. Only the solvers for time-continuous non linear grey model box are considered. There are two options, variable-step or fixed-step solvers. In principle both options are valid, however, Xilinx System Generator (the software used for create the hardware model) requires mandatory a variable-step solver. As the final objective of this work is to create a hardware model of the SCT pendulum, we have considered that the software model uses the same ODE solver than the hardware model. Therefore, only the variable-step solvers will be considered. Such as, the step size can vary from step to step, depending on the model dynamics (in contrast with fixed-step solvers, where the step size is fixed throughout the simulation) it is possible that its simulation time will be shorter than using a fixed-step solver.

The next thing to take into account is the type of problem that will be solved by the ODE solver. There are two different types of problems depending on the accuracy requirements: stiff and non-stiff problems. Remember that a stiff problem consists of differential equations, whose numerical methods used for solving them are numerically unstable, unless its step size was extremely small. This is not our case, because the accuracy requirements that set a bound

on the local truncation error keep the step length well within the region of stability, thereby, we have a non-stiff problem. For this reason there are only three possible ODE solvers useful for us:

A. Variable-step solvers for time-continuous IDNLGREY models:
* **'ode45'** - Runge-Kutta (4,5) solver for **non-stiff problems**.
* **'ode23'** - Runge-Kutta (2,3) solver for **non-stiff problems**.
* **'ode113'** - Adams-Bashforth-Moulton solver for **non-stiff problems**.

In order to choose which one is the best a test was done. First of all, three different idnlgrey_objects were created, with the same properties between them, except the ODE solver. After executing the "*pem*" method, the Adams-Bashforth-Moulton solver was chosen because it obtained the best results in our test. It was the result expected because as is explained in Simulink documentation [16] this solver can be more efficient than others for computationally intensive problems and that is our case.

RelTol - Another important property of Simulation Method for variable-step time-continuous solver is the relative error tolerance (*RelTol*) that applies to all components of the state vector (*x(i)*). The estimated error (*e(i)*) in each integration step satisfies $|e(i)| <= max(RelTol*abs(x(i)), AbsTol(i))$, where *AbsTol* is the absolute error tolerance. The default value is 1e-3, this means 0.1% accuracy, and only positive real numbers are accepted as assignable values for it.

AbsTol - Completing the important properties that can be set on idnlgrey_object, specifically, a property of Simulation Method for variable-step time-continuous solver, it is the absolute error tolerance or threshold error value. This tolerance represents the acceptable error as the value the measured state approaches zero. The default value is 1e-6 and only positive real numbers are accepted as assignable values for it.

SEARCH METHOD

This property set the method used by the iterative search algorithm. There are four different methods to choose. These methods are based on the minimization schemes and are typified as on-line search methods. It is possible to choose between Gauss-Newton type methods (Gauss-Newton and Adaptive Gauss-Newton method), steepest-descent methods (Gradient method) and Levenberg-Marquardt methods (Levenberg-Marquardt method).

In this application, the best search method is the **Levenberg-Marquardt** method because it is valid for nonlinear functions and it is more robust than the others (Gauss-Newton methods and Gradient method) as is shown in [17]. The problem could be the time spent in the search, because the other methods can be a little bit faster if starting parameters are quite reasonable. Levenberg-Marquardt method is used in many applications for solving curve-fitting problems [18]. It is so popular because it can find a solution even if the starting point of the algorithm is very far from the final minimum.

The last property useful for parameters identification is **MaxIter**, maximum number of iteration allowed by execution of "*pem*" method. The default value is 20 and it only accepts positive integer values. It is an important property because sometimes the "*pem*" method could be spent a lot of time iterating, even days, and limiting this value is possible to obtain some results faster. As a rule this property was set to its default value and depending on the observation of the iterations its value could be changed, for instance, if "*pem*" method

execution spends a lot of time iterating, normally, this value is changed to a lower number because we do not like to wait days for obtaining the results. Note that until the end of "*pem*"execution the final values of our model parameters are not accessible.

RESULTS

First of all, we want to call the reader's attention to this section because the great majority of effort spent in this master thesis was doing the parameter identification. As Jin et al. [19] said in their paper a non-linear model identification using a greybox model has its tricky part in the parameter estimation. The reason gives by Jin et al. is that the search for a good fit using training data tend to lead to a increasingly complex model. After months spending on this identification of parameters we can assert they were right.

Next, a list in time order, which enumerates all decisions taken to fix the problems and, finally, to find out the model parameters, is shown:

1. Chirp signals test: using High and Low frequency variation with time
    a. All parameters fixed except b_θ
    b. All parameters fixed except b_θ and b_x
    c. All parameters fixed except friction coefficients μ_S, μ_C, b_θ and b_x
    d. All parameters free
2. Testing with continuously differentiable friction models:
    a. Continuously Differentiable Friction Model: six new parameters.
    b. Exponential Friction Model: two new parameters v_S and γ
        i. All parameters fixed except friction coefficients μ_S, μ_C, b_θ, b_x, v_S and γ
3. Testing with real Controllers signals
    a. All parameters fixed except friction coefficients μ_S, μ_C, b_θ, b_x, v_S and γ
    b. All parameters fixed except μ_S, μ_C, b_θ, b_x, v_S, γ, V_{off} and α.
4. Testing with real Swing Up Controller signal
    a. All parameters fixed except μ_S, μ_C, b_θ, b_x, v_S , γ, V_{off} and α

Unfortunately it is impossible to show every partial result because this master thesis length could be too long. But the most important graphs and decisions taken will be explained in this section.

1. Chirp signal test

The first idea for identifying the parameters of our greybox model consists on applying a chirp signal as input voltage (V_{IN}) because this type of signal is commonly used to characterize dynamical system due to an important feature: its variation in frequency. Remember that a chirp is a signal where its frequency increases/decreases with time, and,

also, that the input signal in our system comes from the controllers block and, a priori, the frequency may be variable depending on the set point fixed by controllers. Therefore, chirp signal will be the first test trying to characterize our dynamical system: the SCT inverted pendulum.

In order to do this identification training data are necessary. Therefore, it is necessary to make an experiment with the real pendulum using a chirp signal as input voltage. To build this signal we have used a hardware circuit, which is implemented in a FPGA into the RAPTOR 2000. This schematic is not shown because of the large extension of documentation. The target chirp signal has a voltage offset of 5V (i.e. zero movement) and an amplitude of 1V peak.

This hardware circuit also allows defining how is the frequency variation of the chirp signal. For this test we have used two different chirp input signal, the first one with a high variation of frequency in time (see Figure II.22), and the second one with a very low variation of frequency (looks like a sinusoidal see Figure II.23), The reason for using these two different type of chirp signals is to know if the model parameters could be find out in both cases or only in one case.

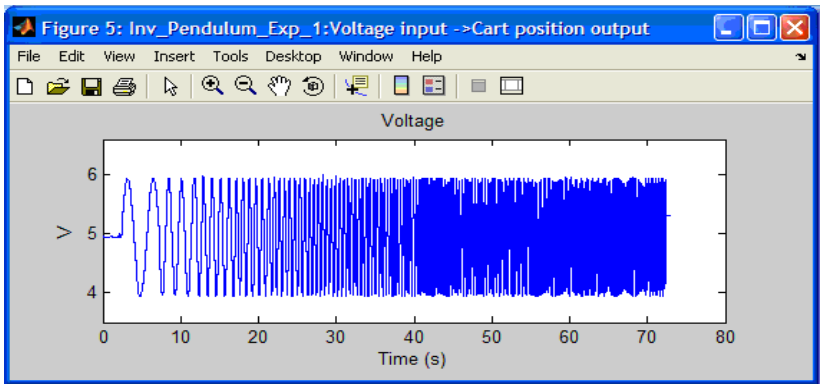


Figure II. 22: Chirp input signal with high frequency variation

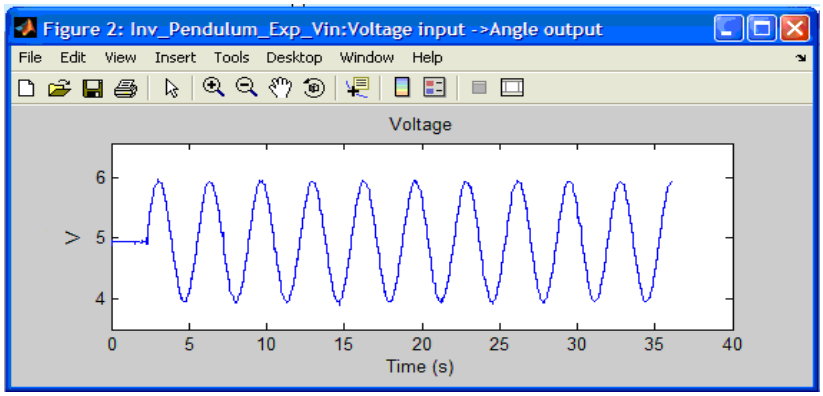


Figure II. 23: Chirp input signal with low frequency variation

Finally, the cart and angle position (system outputs) for each input signal have been stored (see Figure II.24 and II.25) in order to generate the iddata_object, which as was shown in Figure II.21 is necessary to fit outputs of the model (idnlgrey object) to real outputs identifying thus the parameters.

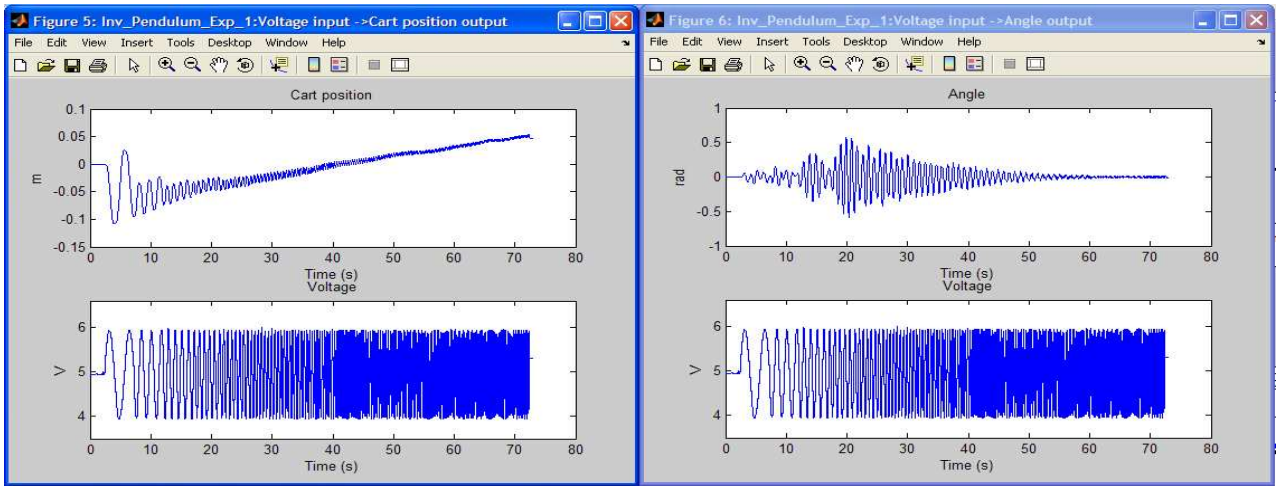


Figure II. 24: Cart (left) and Angular (right) position measured for high freq. variation chirp input

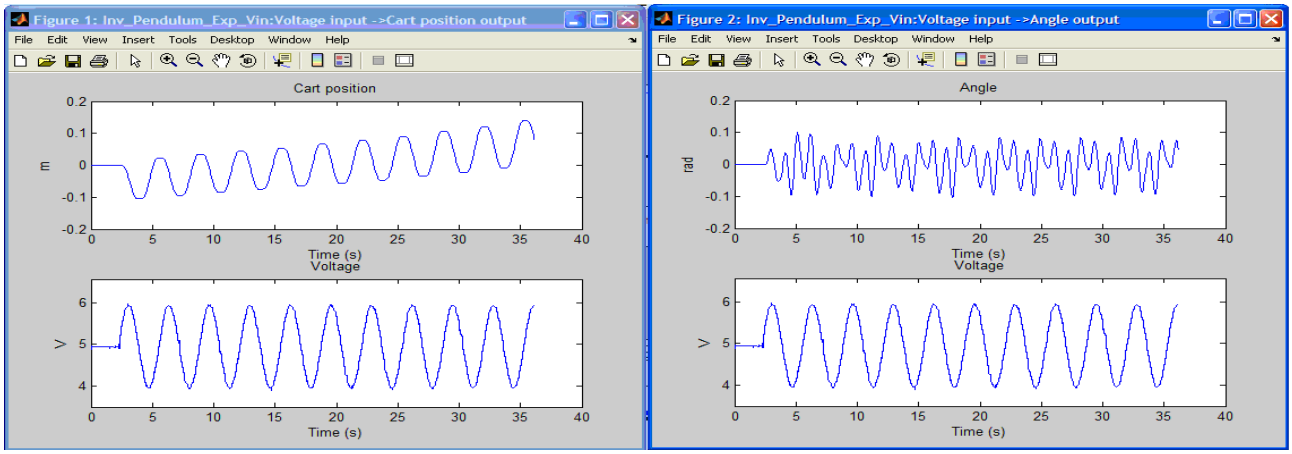


Figure II. 25: Cart (left) and Angular (right) position measured for low freq. variation chirp input

As can be seen at first sight there is a problem with cart position graph in both chirp inputs: the cart position shows an upwards trend (left cart movement). The problem consists on the ideal chirp signal differs a little bit from the measured signal. This trend is because of quantization errors of the DAC, because the input voltage is not symmetric, it spends more time below 5 V than above.

In Figure II.26 the red line represents the ideal chirp signal and the blue line is the measured chirp signal after the DAC block and the bottom subplot shows the DAC error in black line. This DAC error may be interpreted as random variable distributed with mean equals to 0.0601 and a standard deviation of 0.0114.

In Figure II.27 the red line represents the ideal chirp signal and the blue line is the measured chirp signal after the DAC block and the bottom subplot shows the DAC error in blue line and taking the DAC error as random variable, the mean value is equals to 0.0602 (red line) and the standard deviation is 0.0162, which is shown in Figure II.27 as mean + standard deviation (black line) and mean – standard deviation (green line).

This DAC error makes a difference in the input force of about 3N (see Equation II.1) less than the ideal input force and, thereby, the cart position shows an upwards trend (left cart movement).

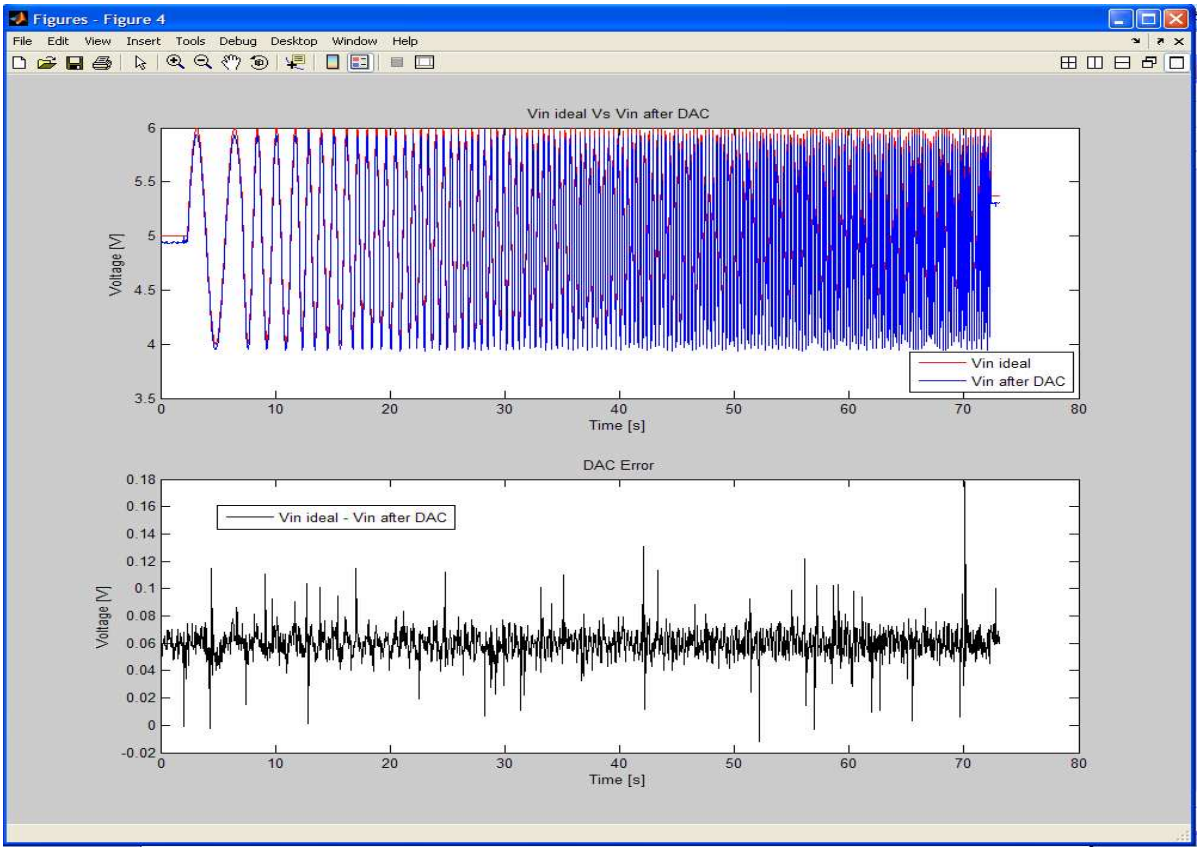


Figure II. 26: V_{IN} ideal Vs after DAC and its difference for high freq. variation chirp input signal

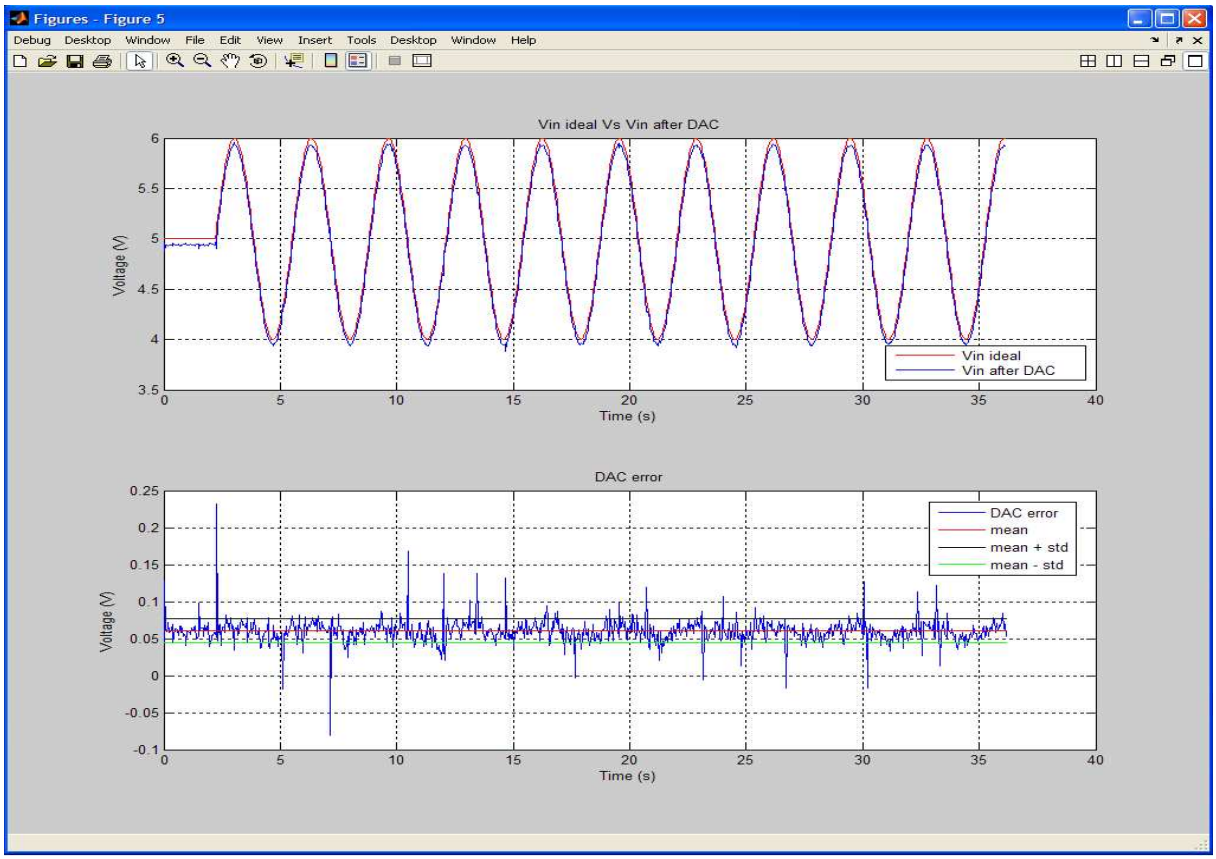


Figure II. 27: V_{IN} ideal Vs after DAC and its difference for low freq. variation chirp input signal

Next step is to start the identification, in Figure II.28 and II.29 are shown the comparison between real pendulum response (black line) and our non linear greybox model response (blue line) for its initial values (before PEM executions). As seen there is a fitting percentage for each output.

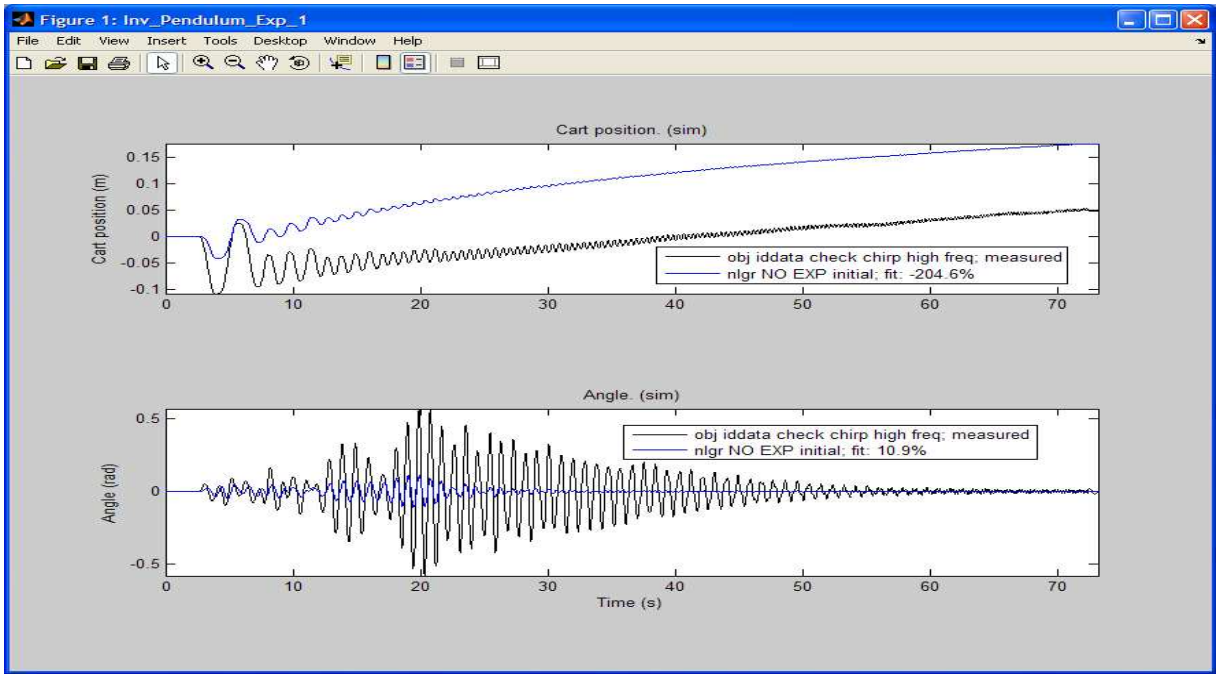


Figure II. 28: Initial fitting for Cart and Angular position with high freq. variation chirp input signal

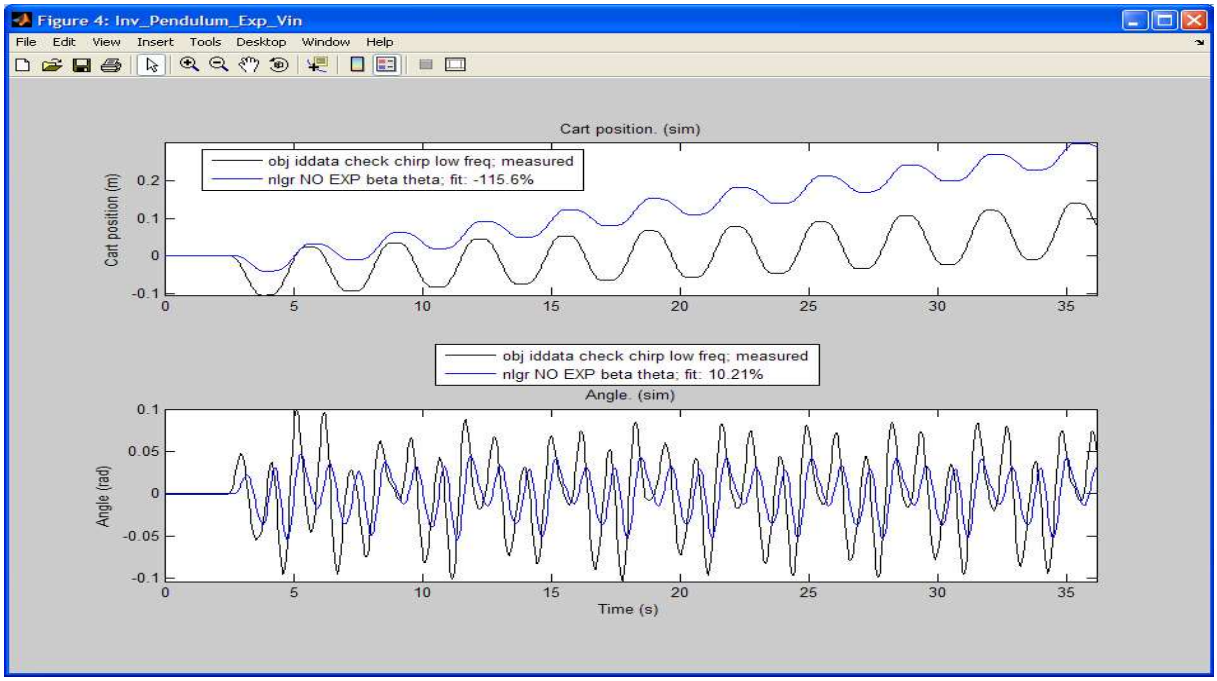


Figure II. 29: Initial fitting for Cart and Angular position with low freq. variation chirp input signal

As is shown in Figure II.28 and II.29 the initial fitting is too far from the desirable target (≈90%), the first idea is that low frequency variation returns better results than high frequency variation because its cart position initial fitting percentage is much better. But it is early to conclude anything.

1. a. All parameters fixed except b_θ

The first idea was to fit the idnlgrey_object, which represents the STC pendulum model, with only one free parameter: the viscous friction coefficient of the rod (b_θ) because it is the only unknown parameter of our model. Unfortunately, as we could see early, this initial idea was impossible to do; thereby, other alternatives were considered in order to fit the idnlgrey_object. As the reader will understand, it is impossible to show every graph or result obtained in this hard way that is the parameter identification of a non linear greybox model.

The first process of parameters identification done in this master thesis corresponds with the case: chirp input signal (high and low frequency variation with time) and all model parameters fixed except b_θ. Note that this process will be repeated for other cases.

1. a. High frequency variation with time

As is shown in Figure II.28, the initial fitting between our model and the real pendulum is really far to be perfect, especially the cart position output with -204.6% of fitting percentage. Our hope consists on executing the PEM method that should find out a better value for parameter b_θ, which should improve this fitting percentage. Unfortunately, after the first execution of the PEM method, which is executed using Levenberg-Marquardt as search method and default values for absolute and relative tolerance and number of maximum iterations, the desirable improvement is really poor, as is shown in the left side of Figure II.30. This first execution finished before reaching the number of maximum iterations (20 is the default value) that means the method converged with the tolerance values defined previously. If we want to do a new execution of PEM method trying to find better fitting, it is necessary to change the absolute/relative tolerance values, in our case we changed them one order of magnitude less. The results obtained after this second execution of PEM method is shown in right side of Figure II.30.

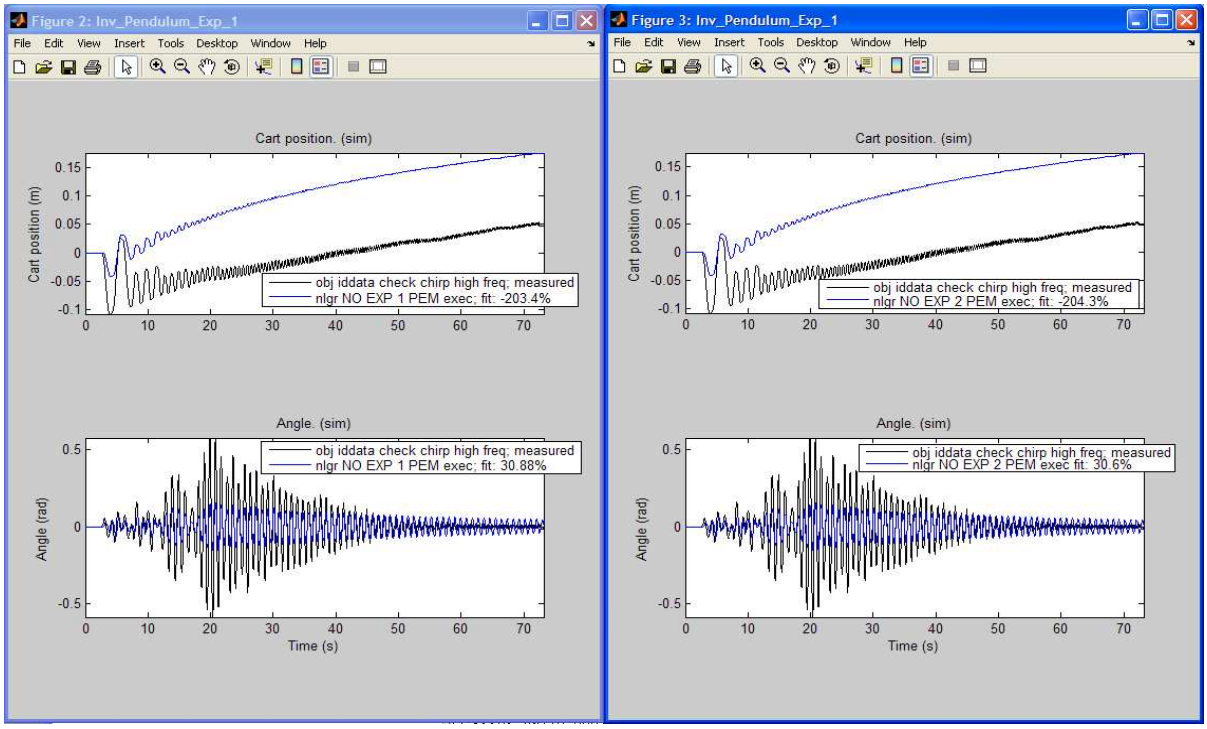


Figure II. 30: Results obtained after 1st (left) and 2nd (right) execution of PEM method for 1.a. High

In order to better see the results obtained it is shown the Table 4, where we can see the free parameters and its values, and, also, the cart and angular position fitting percentages, before and after the executions of the PEM method.

	$b_\theta \ \left(N \cdot s/rad\right)$	$b_x \ \left(N \cdot s/m\right)$	μ_S	μ_C	Cart fit. %	Angle fit. %
Fixed	False	True	True	True	-	-
Initial	0.01	-	-	-	-204.6	10.9
1st exec	0.000554034	-	-	-	-203.4	30.88
2nd exec	0.000561645	-	-	-	-204.3	30.6

Table 4: PEM method results for chirp high frequency variation & all parameters fixed except $\boldsymbol{b_\theta}$

Comparing the results obtained after 1st and 2nd execution of PEM method is possible to conclude that this is not the way for fitting our model because both output percentages (cart and angular position) are worse in 2nd execution than in 1st. Obviously these results improves the initial fitting, especially the angle percentage (from 10% to 30%), but 2nd execution results should be better than 1st and they are not, even get worse from -203.4% to -204.3% in cart position and from 30.88% to 30.6% in angular position. Next step is to probe the case of low frequency variation for the chirp input signal; perhaps these results will be better.

1. a. Low frequency variation with time

As is shown in Figure II.29, the initial fitting between our model and the real pendulum is not quite good but it is better than the initial fitting obtained with the chirp signal with high frequency variation. The angular position percentage is close to 10% that is a little bit worse than this result in chirp with high frequency variation, but the cart position percentage is much better (-115.6% instead of -204.6%).

Next step consists on executing the PEM method that should find out a better value for parameter b_θ, which should improve this fitting. Using default values explained before for the PEM method the results obtained after the first execution are shown in the left side of Figure II.31. As the PEM method converged before reaching the maximum number for iterations fixed by IterMax, for executing again PEM method it is necessary to change tolerance values. Our criterion, which has been established after various tests and checking that it is the best option to find the compromise between the method convergence and the number of iterations to achieve it, consists on decrement both tolerance values one order of magnitude. The results of this second execution are shown in the right side of Figure II.31.

The results obtained are also shown in the Table 5. If the results of the second execution will show better percentages, the process for parameters identification will continue decrementing tolerance (if it will be necessary) and executing PEM method again, but as is shown in Table 5 it is not the case.

	$b_\theta \ \left(N \cdot s/rad\right)$	$b_x \ \left(N \cdot s/m\right)$	μ_S	μ_C	Cart fit. %	Angle fit. %
Fixed	False	True	True	True	-	-
Initial	0.01	-	-	-	-115.6	10.21
1st exec	0.0013466	-	-	-	-114.1	17.84
2nd exec	0.00105337	-	-	-	-114.8	17.92

Table 5: PEM method results for chirp low frequency variation & all parameters fixed except $\boldsymbol{b_\theta}$

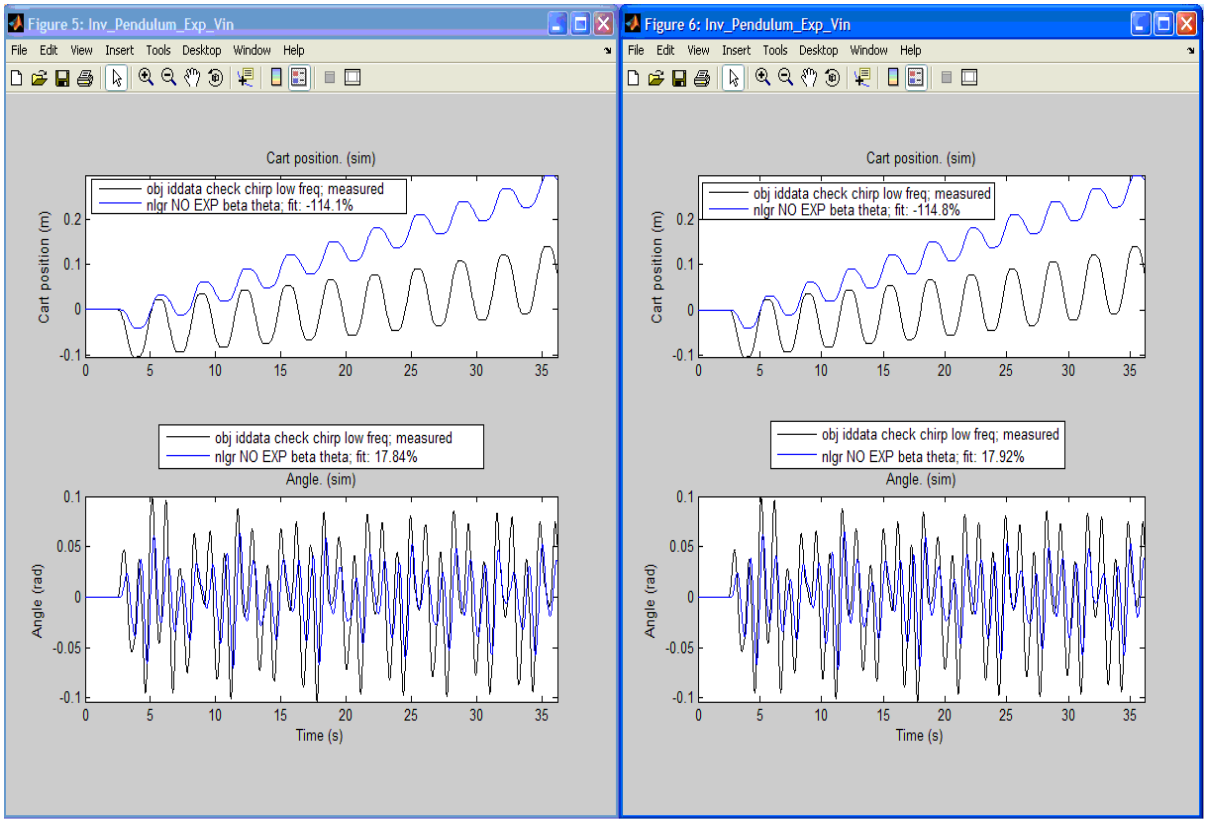


Figure II. 31: Results obtained after 1st (left) and 2nd (right) execution of PEM method for 1.a. Low

The first conclusion is that is impossible to fit tuning only the parameter b_θ, because in both chirp signals the improvement in the fitting is really small, or there is even no apparent improvement but worsening, and the algorithm is converging. To illustrate it, the evolution of this parameter is: from the initial value of 0.01 to 0.00134366 after the first execution of PEM, and to 0.00105337 after the second execution. This means that first the algorithm makes the parameter to grow up, and in the second execution the algorithm makes the parameter decrease its value. Therefore, the minimum is between both values and the fitting improvement is really small.

Next idea was to give a new degree of freedom to the PEM method to try to find a better fitting. This is achieved by changing the value of another parameter (distinct from b_θ) property "Fixed" to *false*, which means to set this parameter as *free* for the identification. In this case, the best parameter to free is the viscous friction coefficient of the cart because, as was shown previously, the main problem appeared in the cart position and this coefficient is the most significant parameter affecting the cart position of the pendulum.

1. b. All parameters fixed except b_θ and b_x

Thereby, the first step consists on changing the value of the parameter property "Fixed" to *false* for the viscous friction coefficient of the cart into the m-file that creates the idnlgrey_object.

Now is possible to begin the process of parameters identification for this new case. Two tests with the high and low frequency variation of chirp input signal will be done again.

1. b. High frequency variation with time

The input voltage and the outputs of SCT pendulum are the same that previously high frequency section. This means that the initial fit still being -204.6% in cart position and 10.9% in angular position as is shown in Figure II.28.

After one execution of PEM method, the fitting improves to -48.31% in cart position and 15.92% in angular position, as is shown in left side of Figure II.32. These improvements confirm that it was a good idea to give another degree of freedom in the identification system, especially, seeing the cart position fitting percentage. Another important thing is to know the new values of the free parameters, which changes because of the fitting method. Using the Matlab command "*present*" we can see that these parameters have been changed to 861.601 Ns/m for b_x and to 0.000294643 Ns/(m·rad) for b_θ.

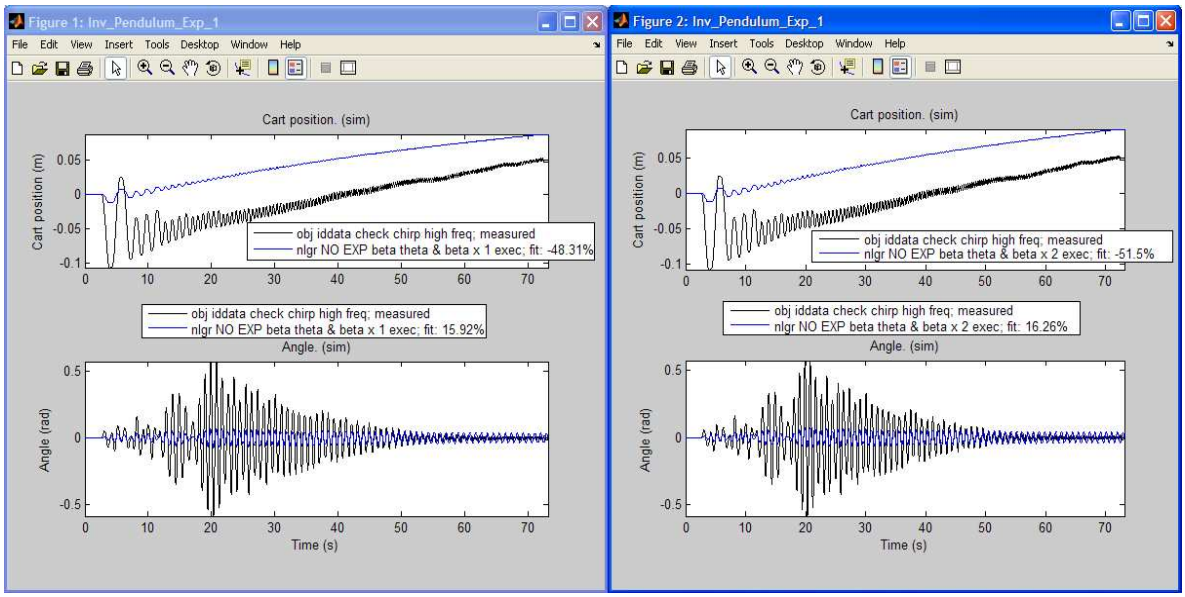


Figure II. 32: Results obtained after 1st(left) & 2nd(right) execution of PEM method for case 1.b.High

After a second execution of the PEM method, previously absolute and relative tolerance have been changed to 1e-7 and 1e-4, respectively, the fitting improves just a little bit as is shown in the right side of Figure II.32. In fact, the new cart position fitting is worse (-51.5%) but the angular position fitting improves to 16.26% and this result has a best cost function than the last one. Both free parameters have been changed to 825.835 Ns/m the viscous cart coefficient (b_x) and to 0.000229558 Ns/(m·rad) the viscous pendulum coefficient (b_θ).

In order to better see the results obtained the Table 6 is shown:

	$b_\theta \ \left(N \cdot s/rad\right)$	$b_x \ \left(N \cdot s/m\right)$	μ_S	μ_C	Cart fit. %	Angle fit. %
Fixed	False	False	True	True	-	-
Initial	0.01	155.1574	-	-	-204.6	10.9
1st exec	0.000294643	861.601	-	-	-48.31	15.92
2nd exec	0.000229558	825.835	-	-	-51.5	16.26

Table 6: PEM method results for chirp high frequency variation & all parameters fixed except $\boldsymbol{b_\theta}$ and $\boldsymbol{b_x}$

We can continue doing more executions of the PEM method (changing the tolerances values) trying to improve the fitting. But it is not the right way to find a good model for our pendulum because, as is shown in Figure II.32, the amplitude of the cart oscillations are decreasing compared with the real cart position. These bad results are also seen at the angular position, remember Table 4 where the fitting percentage for angle is more than 30% and now it is around 16%. These results could improve using a low frequency variation with time chirp input signal.

1. b. Low frequency variation with time

The input voltage and the outputs of the SCT pendulum are the same as in the previous low frequency section. This means that the initial fit still being -115.6% in cart position and 10.21% in angular position as is shown in Table 7.

After one execution of the PEM method, the fitting improves to 15.22% in cart position and 23.6% in angular position as is shown in left side of Figure II.33. Both free parameters have been changed to 720.058 Ns/m the viscous cart coefficient (b_x) and to 0.00190118 Ns/(m·rad) the viscous pendulum coefficient (b_θ). After a second execution of PEM method, previously absolute and relative tolerance have been changed to 1e-7 and 1e-4, respectively, the fitting improves just a little bit as is shown in the right side of Figure II.33. In fact, the new angular position fitting is worse (23.4%) and the cart position fitting improves one tenth to 15.32%. Both free parameters have been changed to 730.27 Ns/m the viscous cart coefficient (b_x) and to 0.00195878 Ns/(m·rad) the viscous pendulum coefficient (b_θ).

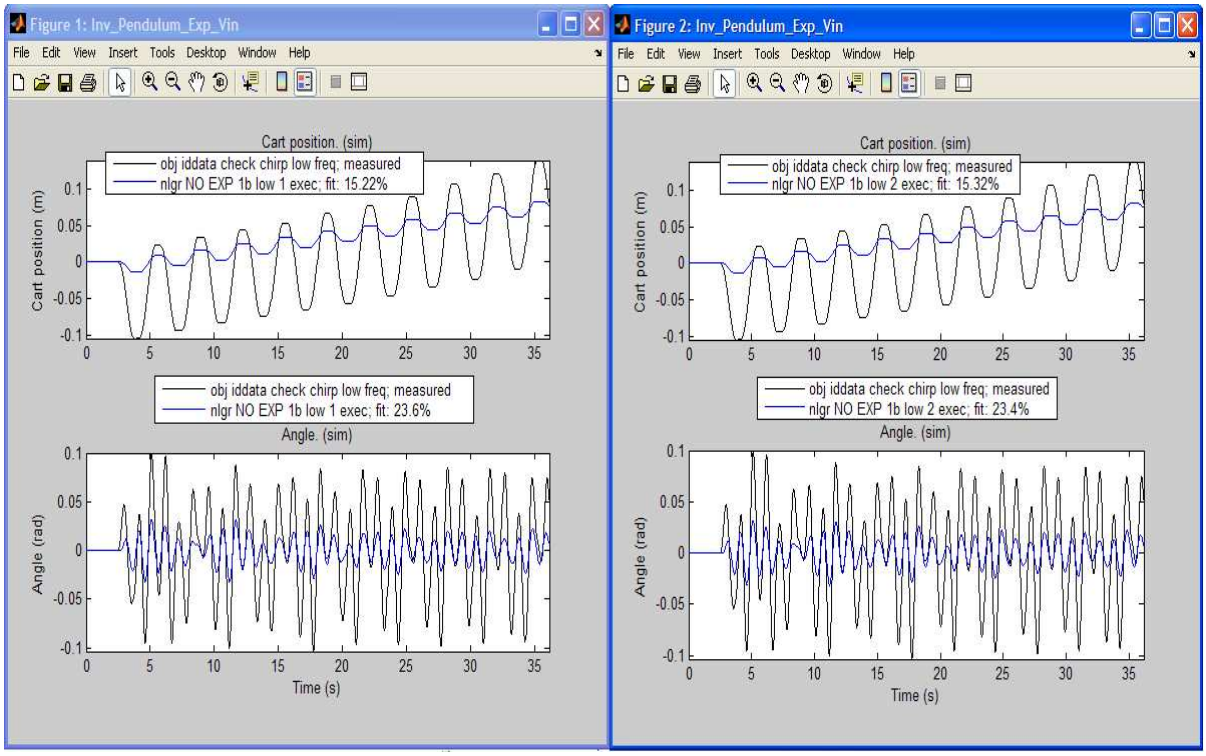


Figure II. 33: Results obtained after 1st (left) and 2nd (right) execution of PEM method for 1.b. Low

In order to better see the results obtained the Table 7 is shown:

	$b_\theta \ \left(N \cdot s/rad\right)$	$b_x \ \left(N \cdot s/m\right)$	μ_S	μ_C	Cart fit. %	Angle fit. %
Fixed	False	False	True	True	-	-
Initial	0.01	155.1574	-	-	-115.6	10.21
1st exec	0.00190118	720.058	-	-	15.22	23.6
2nd exec	0.00195878	730.27	-	-	15.32	23.4

Table 7: PEM method results for chirp low frequency variation & all parameters fixed except $\boldsymbol{b_\theta}$ & $\boldsymbol{b_x}$

We can continue doing more execution of PEM method (changing the tolerances values) trying to improve the fitting, but it is not the right way to reach our objective because the results are far from expected. Going this way we would reach only a small improvement in the fitting percentages and we need a larger improvement.

Next idea was to increase the number of free parameters in our non linear greybox model, specifically, all friction parameters (μ_S, μ_C, b_θ and b_x) because their initial values are probably more inaccurate than the other parameters, remember they have been obtained in experimentally way.

1. c. All parameters fixed except friction coefficients μ_S, μ_C, b_θ and b_x

Setting to false the value of the property "Fixed" for all friction parameters within the m-file that creates the idnlgrey_obj, we can begin the process of parameters identification for this new case. As was stated previously, it is impossible to show every graph obtained trying to find the best parameter values for our non linear greybox model, instead of graphs, the results obtained are going to be shown in tables.

After this explanation we can continue with the tests, high and low frequency variation of chirp input signal.

1. c. High frequency variation with time

The results obtained after the first execution of the PEM method, which are shown in Table 8, are somewhat contradictory because although we found the best fitting percentage for angular position (42.69%), the cart position fitting percentage (-73.4%) is worse than the percentage (-48.31%) obtained for the previous case of high frequency variation (see Table 6). Furthermore, it was the first time that the execution of the PEM method finished because of the maximum iteration number has been reached (after 20 iterations). This means that the PEM method has not converged yet and it needs one or more executions for converging.

Doing a second execution of PEM method but this time without changing tolerances the results obtained were exactly the same of first execution. Even we did a third execution of PEM, in this time changing tolerances because the second one finished before reaching the iteration limit, and the percentages of fitting were again the same.

Using the Matlab command "*present*" we can see the model parameter values, focusing on friction coefficients their values after three execution of PEM method are: Viscous cart coefficient (b_x) 1205.11 Ns/m; viscous pendulum coefficient (b_θ) 0.000938758 Ns/(m·rad); Coulomb friction coefficient (μ_C) 0; and Static friction coefficient (μ_S) 0.0037. These values are quite different compared to initial values, see Table 3, but these results could improve using a low frequency variation with time chirp input signal.

	$b_\theta \ \left(N \cdot s/rad\right)$	$b_x \ \left(N \cdot s/m\right)$	μ_S	μ_C	Cart fit. %	Angle fit. %
Fixed	False	False	False	False	-	-
Initial	0.01	155.1574	0.03709	0.0292	-204.6	10.9
1st exec	0.000938758	1205.11	0.0037	0	-73.4	42.69
2nd exec	0.000938758	1205.11	0.0037	0	-73.4	42.69
3rd exec	0.000938758	1205.11	0.0037	0	-73.4	42.69

Table 8: PEM method results for chirp high frequency variation & all parameters fixed except $\boldsymbol{\mu_S}$, $\boldsymbol{\mu_C}$, $\boldsymbol{b_\theta}$ & $\boldsymbol{b_x}$

1. c. Low frequency variation with time

The results obtained, which are shown in Figure II.34 and Table 9, after the first execution of the PEM method are unexpected because the model outputs (cart and angular position) are zero until 15s, from this time to the end are quite close to real outputs. In fact, the fitting percentage for cart position is 26.43% that is the best result obtained so far, although for angular position it is 13.58%. The explanation for this behaviour of non-movement of the pendulum is because until 15s there is not an input voltage quite enough high to exceed static friction force. As now the static friction coefficient is a free parameter for the PEM method, it increases this value to 0.0542; thereby, the static friction force is increased and there is not pendulum movement in the early 15s.

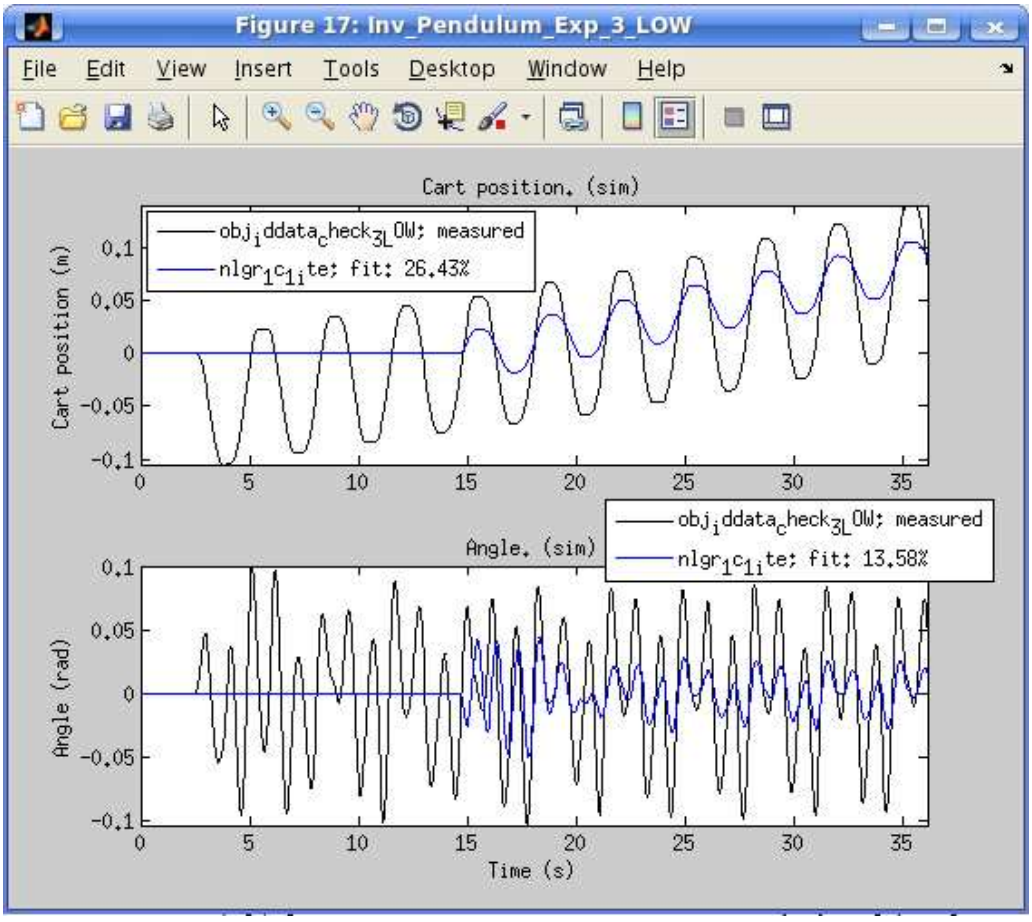


Figure II. 34: Results obtained after 1st execution of PEM method for 1.c. Low

	$b_\theta \ \left(N \cdot s/rad\right)$	$b_x \ \left(N \cdot s/m\right)$	μ_S	μ_C	Cart fit. %	Angle fit. %
Fixed	False	False	False	False	-	-
Initial	0.01	155.1574	0.03709	0.0292	-115.6	10.21
1st exec	0.00829059	556.883	0.0542	0.0175709	26.43	13.58

Table 9: PEM method results for chirp low frequency variation & all parameters fixed except $\boldsymbol{\mu_S}$, $\boldsymbol{\mu_C}$, $\boldsymbol{b_\theta}$ & $\boldsymbol{b_x}$

Although for the PEM method, which only takes into account the cost function, this behaviour is better than previous results we can assert that this is not the right way to find our fit model. Last chance for finding a good model using chirp input signal will be with all model parameters free.

1. d. All model parameters free

In this case the results are the same for both chirp input signal (high and low frequency variation) because their model outputs are zero all the time. Therefore, the fitting percentages are close to 0%. As we said previously, for PEM method it is an improvement from the initial values, at least for cart position, but obviously it is not for us. The explanation is easy: there is too much freedom (thirteen degrees of freedom, one degree for each free parameter) for finding a fit model.

At this point it is necessary to change something in our model because obviously it is not working as we expected. After a thorough analysis we found that our model has a big problem with discontinuities due to our friction model. Paying attention to Equation II.12 and Equation II.13 the change from static friction to dynamic friction force and vice versa is discontinuous because as is shown in Equation II.33 the value for $f'_{FRIC\_X}(t)$ force is obtained depending on cart velocity value and when this value changes from zero to another value a hop is produced in the $f'_{FRIC\_X}(t)$ force.

These hops produced in $f'_{FRIC\_X}(t)$ force introduce discontinuities in our model and, as is known, the real world is continuous. Furthermore, these discontinuities generate problems to solve the differential equations of our model because the discontinuities make functions non differentiable. Therefore a continuously differentiable friction model is needed.

2. Testing with continuously differentiable friction models

Our first idea for applying a continuously differentiable friction model was based on [20]. The main reason for choosing this model was that it is used by Matlab Simulink for modelling the friction. Unfortunately, for reason that will be explained later, it was impossible to apply in our SCT inverted pendulum model.

The second and definitive idea for a continuously differentiable friction model was found in [14], where appears a continuously differentiable friction model based on exponential functions.

2. a. Continuously Differentiable Friction Model: six new parameters

This friction model is based on hyperbolic tangents, which are continuous functions:

$$f(\dot{q}) = \gamma_1\left(\tanh(\gamma_2\dot{q}) - \tanh(\gamma_3\dot{q})\right) + \gamma_4 \tanh(\gamma_5\dot{q}) + \gamma_6\dot{q} \quad \text{(II.102)}$$

Where $\dot{q}$ in our case is the cart velocity and γ_i (for i = 1,...,6) denote unknown positive constants.

This new friction model is related to our previous model in this way:

$$\mu_S = \gamma_1 + \gamma_4 \quad \text{(II.103)} \qquad \mu_C = \gamma_4 \tanh(\gamma_5\dot{q}) \quad \text{(II.104)} \quad \text{and} \quad b_x = \gamma_6 \quad \text{(II.105)}$$

The problem lies in identifying these new six parameters. It is also a non linear greybox model, so there is a subsystem to identify before identifying the whole system. The training data necessary to conduct this six parameters identification consists on a slip speed as input and friction force as output, where this slip speed input is swept from a negative to positive value in a ramp-type manner because when this velocity crosses zero we will be able to observe the change between static and dynamic friction (see Figure II.35).

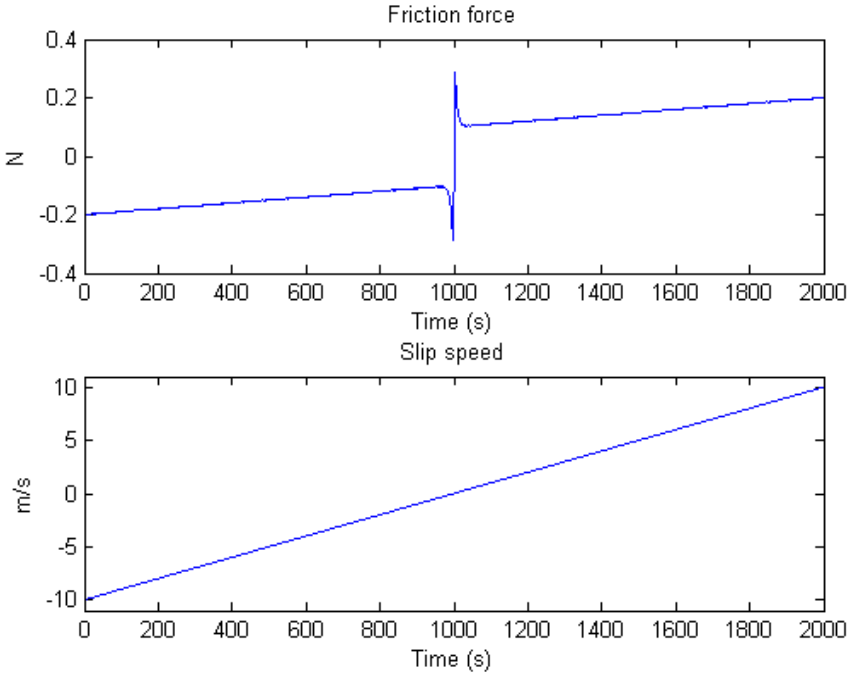


Figure II. 35: Training data necessaries for identifying 6 new parameters of continuously differentiable friction model

Unfortunately, these training data were impossible to obtain because the experiment necessary requires a velocity configuration instead of torque in our Power Driver Controller and it is a really hard extra work for this master thesis. Thereby, we thought of a different idea for finding a continuously differentiable friction model.

2. b. Exponential Friction Model: two new parameters v_S and γ

The new idea came from reading the Sue Ann Campbell et al. paper [14] where is demonstrated that the transition of cart from static to dynamic event is produced continuously. They propose to add an exponential factor in their friction formula to smooth this transition. Therefore, the new formula for cart friction, instead of Equation II.33, is:

$$f'_{FRIC\_X}(t) = \begin{cases} f_{STATIC\_X}(t) \leftrightarrow \dot{x} = 0 \\ -\left(\mu_C + (\mu_S - \mu_C) e^{-\left(\frac{|\dot{x}|}{v_S}\right)^{\gamma}} \right) F_N \operatorname{sgn}(\dot{x}) \leftrightarrow \dot{x} \neq 0 \end{cases} \quad \text{(II.106)}$$

Where appear two new terms: Stribeck velocity (v_S) and form factor (γ).

Stribeck was the first one that observed that for low velocities, the friction is decreasing continuously with increasing velocities and not in a discontinuous way. This phenomenon of a decreasing friction at low, increasing velocities is called the Stribeck friction or effect. Therefore, the new friction model including Static, Coulomb, Viscous and Stribeck friction is shown in Figure II.36.

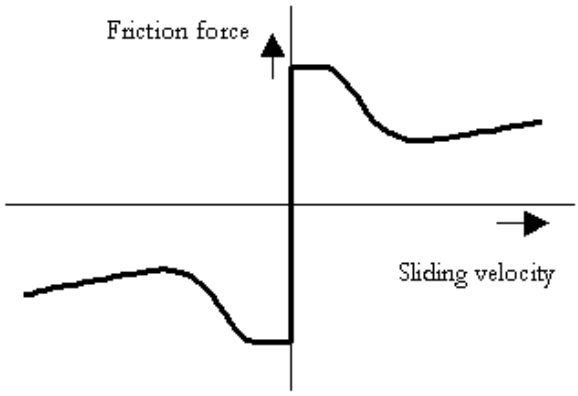


Figure II. 36: Friction model including Static, Coulomb, Viscous and Stribeck friction (see [21])

Stribeck velocity denotes the sliding velocity where only a 37% of the static friction is active. In other words, small values of v_S give a fast decreasing Stribeck effect and large values of v_S give a slow decreasing Stribeck effect. Form factor is also used to model this continuously friction, in this case, large values of γ give a fast decreasing and small values give a slow decreasing Stribeck effect.

2. b. i. All parameters fixed except friction coefficients μ_S, μ_C, b_θ, b_x, v_S and γ

Due to this new friction model some changes are needed in our m-files used for generating the idnlgrey_object. For instance, it is necessary to replace the old formula that obtains cart friction by the new one (see Equation II.106) into the m-file that describes the model ("*inv_pendulum_m.m*"), it is also necessary to include as this file input arguments the two new parameters (v_S and γ).

Therefore, it is necessary to declare these two new parameters into the m-file that generates the idnlgrey_object ("*creation_idnlgrey_object.m*") setting their initial values (taken from [14]) and setting their "Fixed" property to *false* such as μ_S, μ_C, b_θ, b_x (remember the rest of parameters are fixed).

After creating this idnlgrey_object the identification can start. Using high and low frequency variation of chirp input signal for identifying the model parameters, and after some PEM executions, the best results obtained are shown in Figure II.37.

This new friction model appears to be a step in the right direction towards the identification of our system parameters because the fitting percentages obtained, at least for low frequency variation signal, are very good: 88.91% for cart position and 80.89% for angular position. However, the fitting percentages for high frequency variation chirp input signal, which are -58.97% for cart position and 43.29% for angular position, are not enough for us even they are the best result ever for this type of chirp signal.

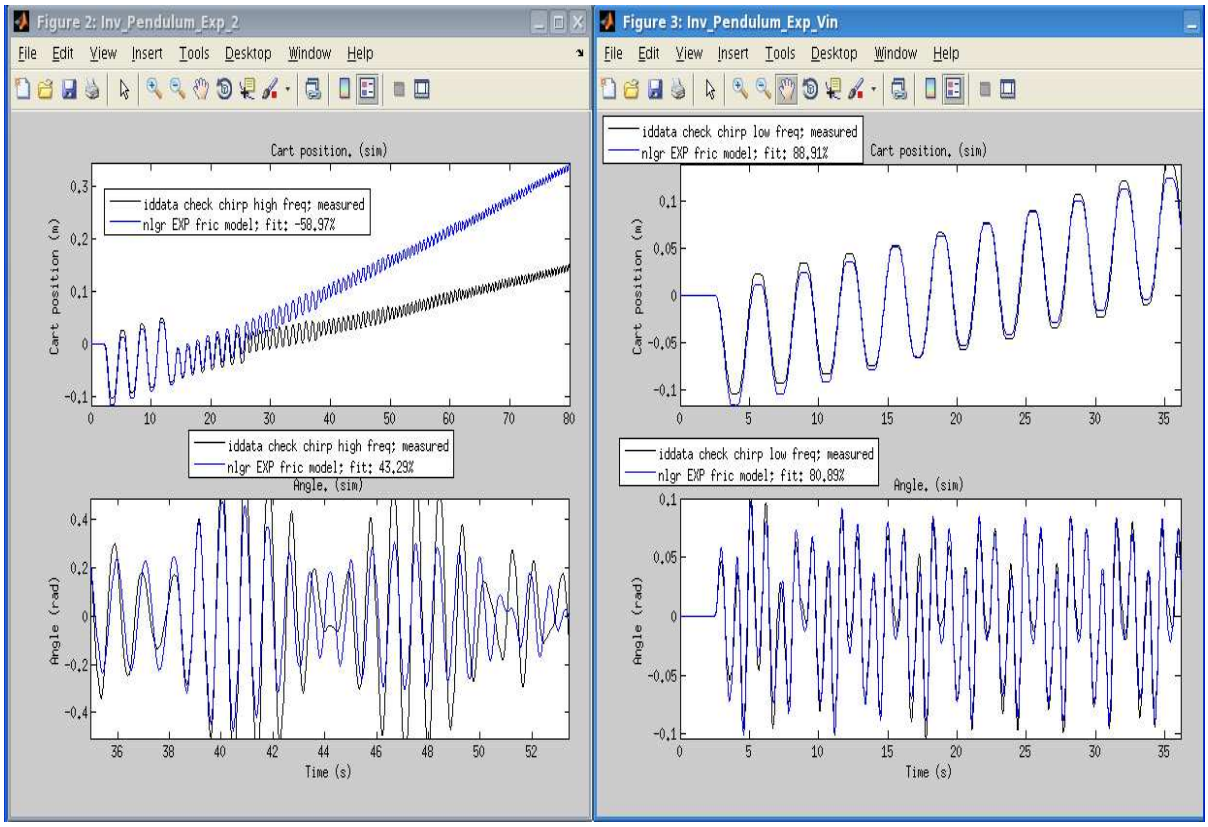


Figure II. 37: Results obtained for 2.b. i. using High (left) and Low (right) frequency variation of chirp signal

	$b_\theta \ \left(N \cdot s/rad\right)$	$b_x \ \left(N \cdot s/m\right)$	μ_S	μ_C	$v_S \ \left(m/s\right)$	γ	Cart %	Ang %
Fixed	False	False	False	False	False	False	-	-
Initial	0.01	155.157	0.0371	0.0292	0.105	2	-204.6	10.9
1st ex	0.0021	1891.89	0.0068	0.0157	0.0589	39.731	-58.97	43.29

Table 10: PEM method results for chirp high frequency variation using an Exponential Friction Model with all parameters fixed except μ_S, μ_C, b_θ, b_x, v_S and γ

	$b_\theta \ \left(N \cdot s/rad\right)$	$b_x \ \left(N \cdot s/m\right)$	μ_S	μ_C	$v_S \left(m/s\right)$	γ	Cart %	Ang %
Fixed	False	False	False	False	False	False	-	-
Initial	0.01	155.1574	0.03709	0.0292	0.105	2	-204.6	10.9
1st ex	0.0086971	894.405	0.180673	0.01314	0.03472	1.1384	88.91	80.89

Table 11: PEM method results for chirp low frequency variation using an Exponential Friction Model with all parameters fixed except μ_S, μ_C, b_θ, b_x, v_S and γ

From this point, the input signals used for the parameter identification will be replaced by the real input signals supplied by the Controllers block because the results obtained using the chirp input signals are far from ideal. Thereby, the next idea is to test using the real input signal with intentions that our model parameters will be better adjusted.

As a summary of the tests for parameter identification using the chirp type signals, the Figure II.38 and II.39 are shown. In these graphs you can see the fitting percentages for the cart and angular position obtained in the chirp signal tests performed so far (cf. subsections 1.a, 1.b, 1.c, 1.d and 2.b including the inital fitting percentages named *init* in Figure II.38 and II.39).

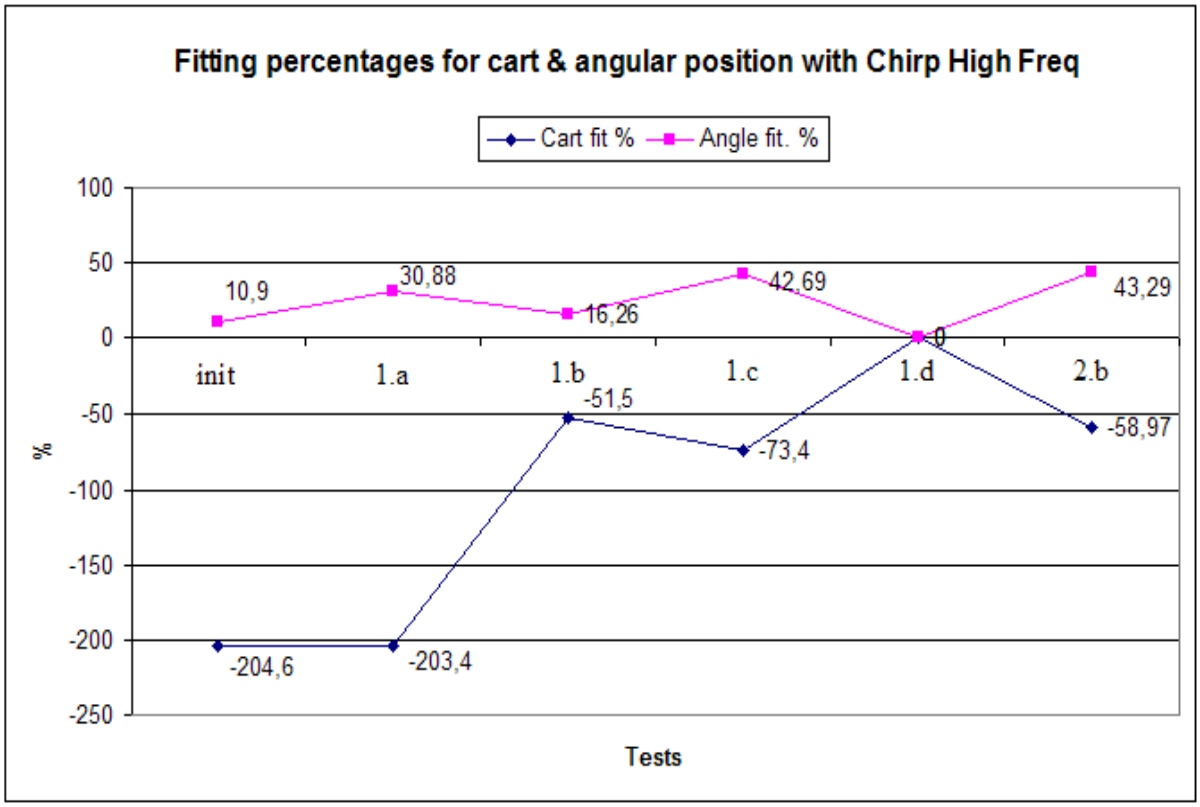


Figure II. 38: Fitting percentages for cart & angular position obtained using the chirp signal with high frequency variation in each test.

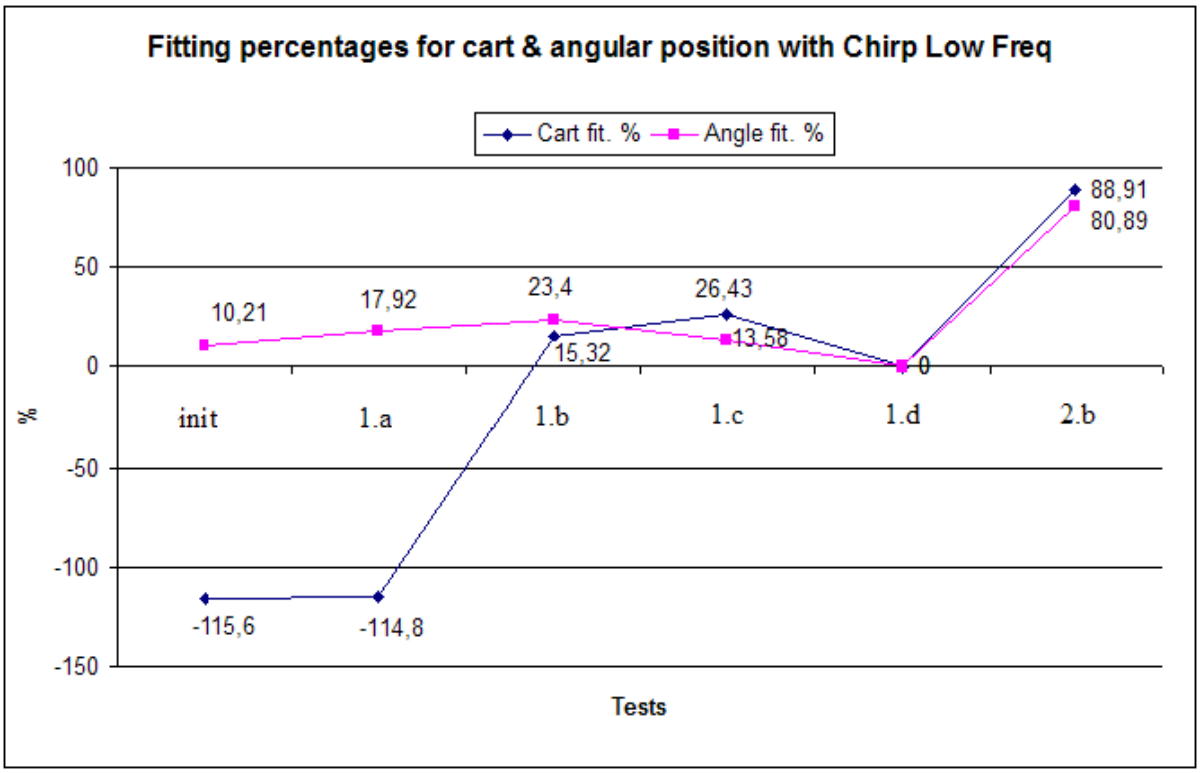


Figure II. 39: Fitting percentages for cart & angular position obtained using the chirp signal with low frequency variation in each test.

Seeing Figure II.38 and II.39 we can conclude that the characterization of our dynamical system using chirp signals, which have the important feature of its variation in frequency, is not enough because of the maximum values of the fitting percentages. But we can not base the choice of parameters approach focused exclusively on the values of these percentages (at least when they are below 70%) because although mathematically 1c test results are better than 1.a test, we know that are much worse looking at the graphs (Figures II.34 and II.31). Another example of this is found in the test 1.d.

Although these tests appear to have been useless, they have not been, because these tests have made us realize the need for a model of continuous and differentiable friction, and also, that some model parameters that we considered as fixed must be free to adjust the model, such as μ_S, μ_C, and b_X .

We also emphasize that the process of characterization of the SCT inverted pendulum gives better results using signals with low frequency variation with time than using high frequency variation with time signals. The reason probably is that the input signals with high frequency variation used in our parameter identification process have a too wide frequency spectrum for our system. Thereby, next question to answer is: Is the real pendulum input signal, which is supplied by Controllers block, a low frequency variation type? To know the answer a new test is needed.

3. Testing with real Controllers signal

First step was to do an experiment with STC inverted pendulum system storing Controllers signal (system input), cart and angular position (system outputs), which are needed for creating the new iddata_object. In Figure II.40 and II.41 are shown the experiment results.

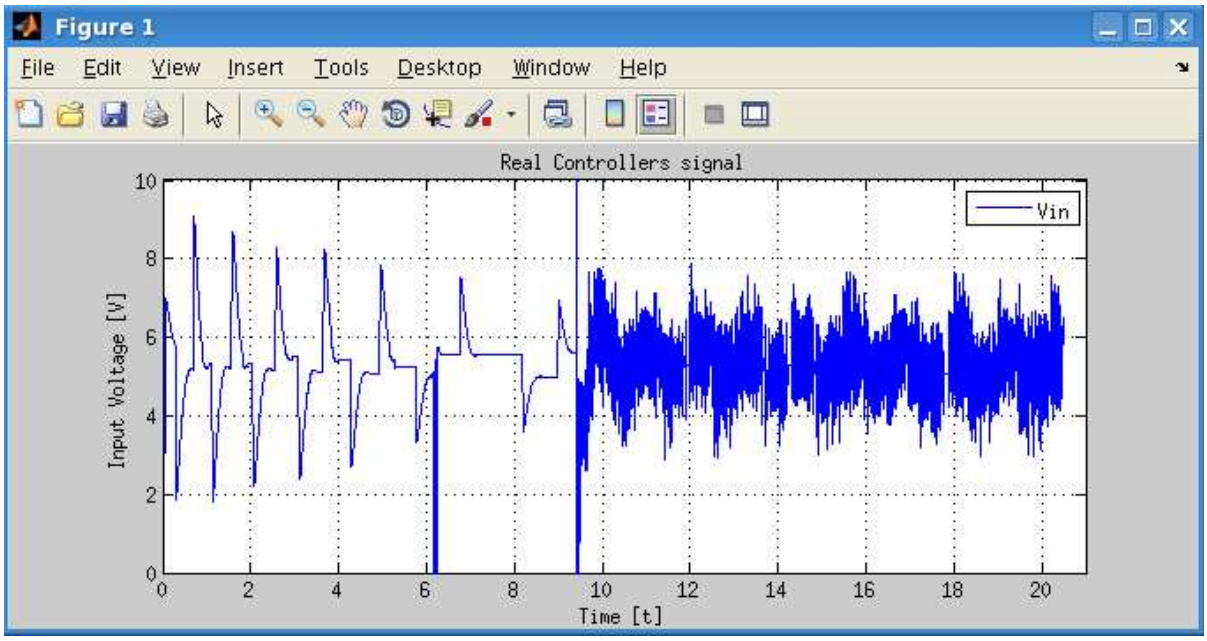


Figure II. 40: Input voltage supplied by Controllers block.

As seen in Figure II.40 until roughly the 9th second the input voltage signal is given by Swing Up Controller and from this 9th second the input signal is provided by Balance Controller (noisy signal). In fact, to be precise, there is a very short period of time (at 6th second) where the input voltage signal is given by the Balance Controller. It is a failed attempt to balance the pendulum.

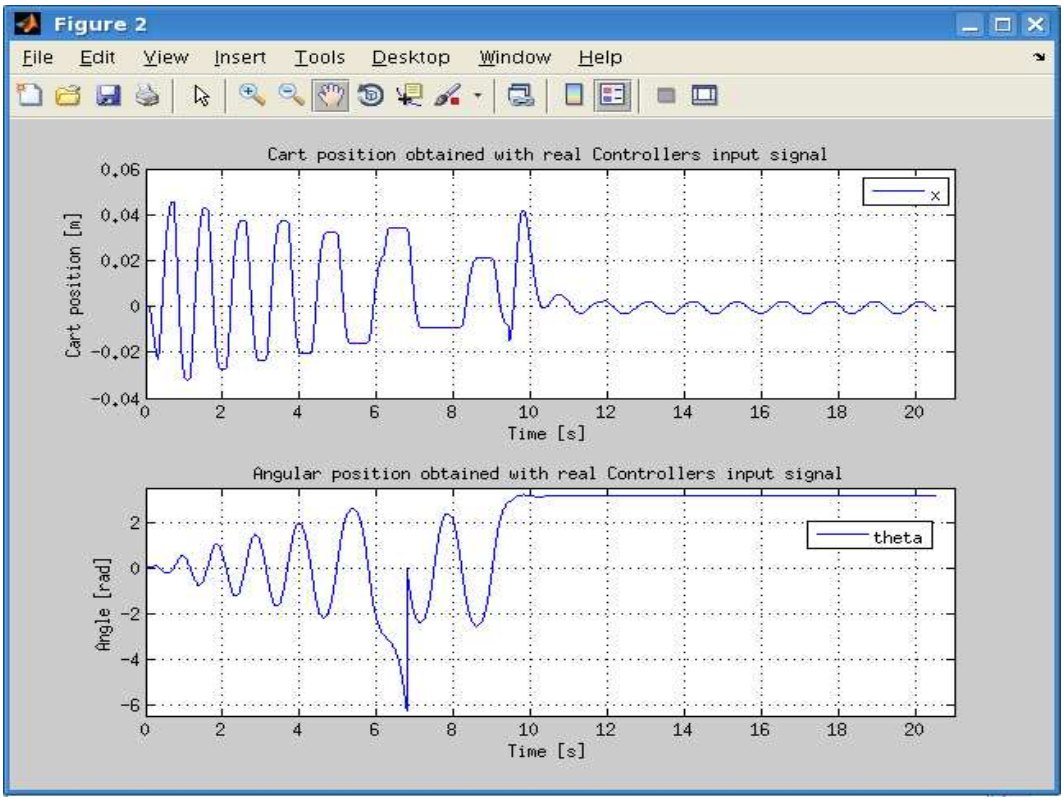


Figure II. 41: Cart and angular position obtained using Controllers block.

Looking angular position in Figure II.41 we can corroborate that from this 9th second the pendulum is balanced in inverted position ($\theta = \pi$). Note that at the 7th second the pendulum makes a loop, which is shown in Figure II.41 when angular position decrease more than -2π and it is instantly redefined as 0. This loop is produced because of the failed attempt to balance the pendulum at the 6th second, when angular position is close to $-\pi$. In fact, there is a quick switching between controllers as seen in Figure II.40 just after the 6th second, where appears the noisy Balance Controller signal which values oscillate very fast around five volts.

Next step is to start identification using the same idnlgrey_object of section *2.b.i.* but with the new iddata_object, which represents our system outputs using real Controllers signal as input, instead of the old iddata_object, which represents our system outputs using Chirp signal as input.

3. a. All parameters fixed except friction coefficients μ_S, μ_C, b_θ, b_x, v_S and γ

After some PEM executions that required many hours (even days) the results, which are not shown, were not entirely satisfactory, especially from 9th second when balance controller is working. The most important problem is the trend of the cart position, at first sight it could be DAC error but looking the input voltage supplied by Controllers block (Figure II.40) it is not symmetrical about 5V. In fact, doing the mean for this input it is 5.3V. This asymmetric behaviour is not analyzed in this master thesis. For solving this problem next idea is to increment the number of free parameters in our model including V_{off} and α because the values of these parameters are directly related to cart movement (see Equation II.1). Remember that these parameters were modelled using a linear simplification, which was applied to all the mechanical blocks of our system except for the pendulum block (i.e. Power Driver, Motor AC, Timing Belt and Spindle block) and, therefore, their initial values are probably not the most reliable due to this simplification.

Therefore, next test was setting the "Fixed" property of these parameters (V_{off} and α) to *false*.

3. b. All parameters fixed except μ_S, μ_C, b_θ, b_x, v_S, γ, V_{off} and α

In this case, the results obtained after identification process are quite good as is shown in Figure II.42. The trend of cart position practically has been disappeared due to parameter V_{off} and the fitting percentage has been improved from its initial value in both outputs (see Table 12), but it remains lower than expected.

Of course, these results are much better than previous (excluding the results of test 2.b for chirp input signal with low frequency variation with time) at least until 9th second, but the percentages are very poor. This is a clear example of why graphics should be observed, and not only the mathematical percentages, when analyzing the results obtained by the PEM method. In addition, it also shows how complex it was to perform the characterization of the STC pendulum.

After a deep analysis, we realized that it is practically impossible to obtain a high fitting percentage for angular position using this training data, because from 10th second the pendulum is balanced. Thereby, if our model can not balance at exactly the same time the rest of time (from 10th second to the end) the input signal is given by Balance Controller and our model it is not balanced, therefore, the fitting percentage gets worse as is shown in Figure II.42.

	b_θ $\left(N \cdot s/rad\right)$	b_x $\left(N \cdot s/m\right)$	μ_S	μ_C	$v_S\left(m/s\right)$	γ	V_{off} (V)	α $\left(N/V\right)$	Cart %	Ang %
Fix	False	False	False	False	False	False	False	False	-	-
Init	0.0087	894.40	0.1807	0.013	0.0347	1.138	5	51.52	-130.34	-84.89
Fin	0.0071	889.76	0.0363	0.108	0.8435	0.916	5.217	258.6	-75.15	-20.46

Table 12: PEM method results for real controllers signals using an Exponential Friction Model with all parameters fixed except μ_S, μ_C, b_θ, b_x, v_S, γ V_{off} and α

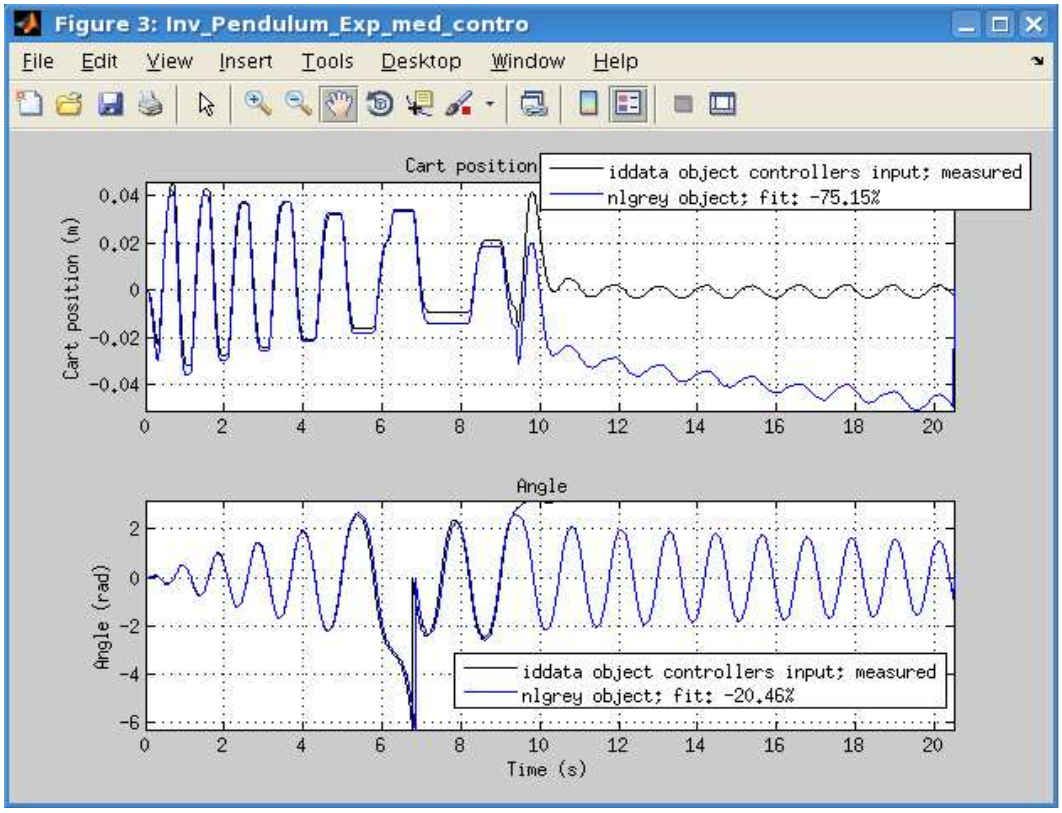


Figure II. 42: Results using real controllers input and nlgrey_object with μ_S, μ_C, b_θ, b_x, v_S, γ, V_{off} and α free.

The next idea is to adjust our model using only the first ten seconds of the input signal, which corresponds to the signal provided by the Swing Up Controller. The reason is clear, our model may be adjusted a lot (the fitting percentages will reach 90%) and, although using these training data our model may not be balanced this does not mean that our model is far from the real pendulum system (as it seems looking at Figure II.42).

The best way to check this is to see if our model implemented in hardware (see chapter 4), when the feedback loop between our SCT pendulum model and the Controllers block is operating (see Figure IV.16), the Controllers block can balance the pendulum in inverted position. Note that if our model is very different it is impossible to be balanced by the controllers without modifying them, because they have designed specifically for the real pendulum.

4. Testing with real Swing Up Controller signal

Generating an iddata_object with input voltage supplied by Swing Up Controller until 7th second and using the nlgrey_object obtained as result in the last test, which has all parameters fixed except μ_S, μ_C, b_θ, b_x, v_S, γ, V_{off} and α, the identification process can start. After some PEM execution the results obtained are shown in Table 13 and Figure II.43.

	b_θ $\left(N \cdot s/rad\right)$	b_x $\left(N \cdot s/m\right)$	μ_S	μ_C	v_S $\left(m/s\right)$	γ	V_{off} (V)	α $\left(N/V\right)$	Cart %	Ang %
Fixed	False	False	False	False	False	False	False	False	-	-
Init	0.007085	889.761	0.0363	0.108	0.843	0.916	5.217	258.61	61.37	83.87
1st ex	0.005438	886.842	0.0639	0.127	0.894	0.938	5.224	259.53	70.48	89.62
Final exec	0.001288	877.3132	0.1036	0.133	0.954	1	5.265	260.79	82.26	94.66

Table 13: PEM method results for Swing Up Controller signal using an Exponential Friction Model with all parameters fixed except μ_S, μ_C, b_θ, b_x, v_S, γ V_{off} and α

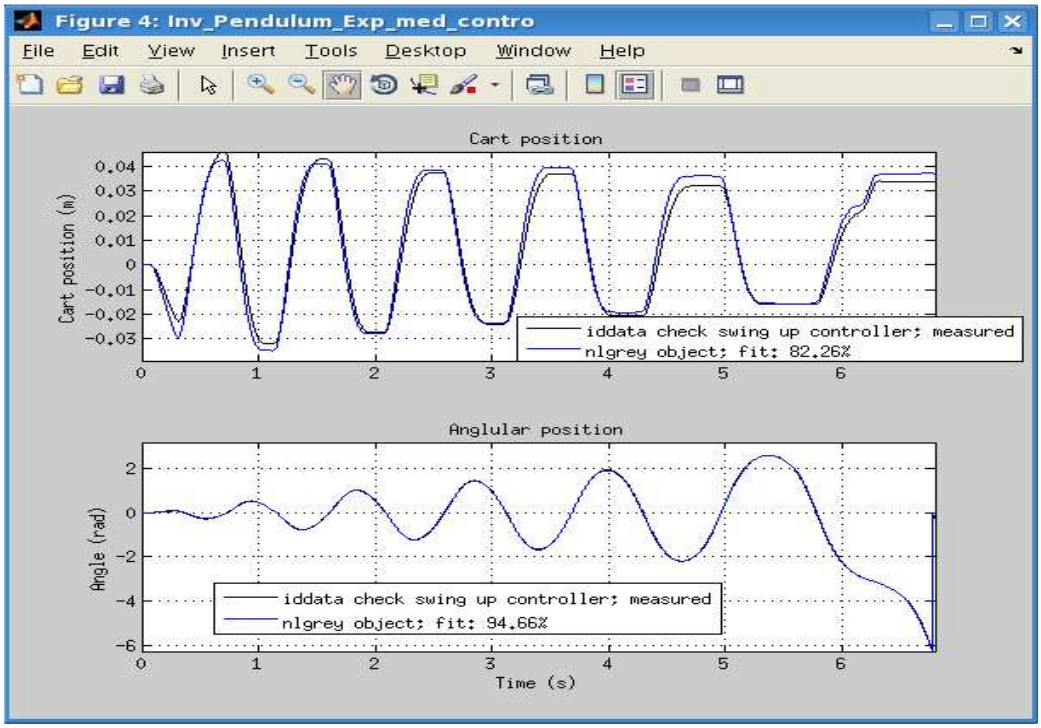


Figure II. 43: Results for SwingUp Controller input and nlgrey_object with μ_S, μ_C, b_θ, b_x, v_S, γ, V_{off} and α free

These results are the best so far achieved and their fitting percentages reach satisfactory levels; therefore, the parameters identification section can be finished. As stated previously, it was impossible to find an nlgrey_object that was balanced at same time that real pendulum because our model simplifies some parts of the real system such as: Power Driver, AC motor, Timing Belt and Spindle block, which were replaced by a linear model; and also the friction model, which remains a model simplified what really happens to the friction in the pendulum.

Thereby, the model found in this section is not exactly the same that SCT inverted pendulum, but it is a simplified and fairly accurate model that we were looking for (see section 1.1.Purpose). And, as stated previously, it will be demonstrated in the chapter 4, when the feedback loop between our SCT pendulum model and Controllers block will be operating and the Controllers block will be able to balance our pendulum model in inverted position easily.

As conclusion of this parameter identification process we can assert that using input signal with low frequencies it works better than using high frequencies, as was demonstrated with chirp signal test. Furthermore, it is better for the process of identification to have the least number of degrees of freedom, which are represented by free parameters, possible, because setting an excessive number of free parameters to the PEM method to adjust our model can distance it from the reality. Note that the solution returned by the PEM method is based on mathematics and, as was demonstrated at the tests using chirp input signals, sometimes a better value in the fitting percentages does not mean a better adjusted model (c.f. the 1.d test). Another important point to consider in the identification process for the parameters of our non linear greybox model is to have a differentiable model because as was demonstrated the fitting percentages are much improved.

The last conclusion is that for finding out the parameter values of our system it is necessary to use as training data the same input signals that are used in the real pendulum, which are provided by the Controllers block. It was important to include as free parameters V_{off} and α because those are the parameters that model, in a simplified way, the blocks of Power Driver, Motor AC, Timing Belt and Spindle. And, as was stated, the initial values of these parameters are largely responsible for the observed differences between our model and the real pendulum.

Finally the parameter values found out are shown in Table 14:

Parameter	Meaning	Unit	Type	Value
g	Gravity acceleration	m/s^2	K	9.80665
L	Distance from pivot to the centre of mass of rod	m	K	0.138867
I	Moment of inertia of the pendulum	$Kg \cdot m^2$	K	6.21034e-3
b_x	Viscous coefficient of the cart	$N \cdot s/m$	K	877.31327
b_θ	Viscous coefficient of the pendulum	$N \cdot s/rad$	U	0.0012889
M	"Virtual" mass of the cart (including moments of inertia)	Kg	K	103.205
m	Mass of the pendulum (rod + bob)	Kg	K	0.581
μ_C	Coulomb friction coefficient	-	K	0.1331798
μ_S	Static friction coefficient	-	K	0.1035951
α	Voltage to force conversion factor	N/V	K	260.78841
v_S	Stribeck velocity	m/s	U	0.9539722
γ	Form factor	-	U	1
V_{off}	Offset voltage	V	K	5.264616

Table 14: Final parameter values for STC inverted pendulum model

There are some parameter values that could be surprising to the reader, especially if they are compared to initial values (see Table 3). For instance, the viscous coefficient of the cart (b_x) is over five times its initial value this means the cart is braked five times much stronger but by the way the cart is accelerated five times faster, how is demonstrated looking to final value of parameter α (voltage to force conversion factor) that has been incremented five times (from 51.525 to 260.8). Therefore, the initial value of α parameter is wrong and, due to it, the initial value of b_x is also wrong, but final values of these parameters are right because the proportion between them is right as is described in Equation II.98.

Final values for viscous coefficient of the pendulum (b_θ), Stribeck velocity (v_s) and form factor (γ) are in principle correct, because their initial values were taken from bibliography only as started point for identification process. Static (μ_S) and Coulomb (μ_C) friction coefficient are a little bit higher than their initial values, but the most significant change is that dynamical friction coefficient is higher than static friction coefficient, as is known it is physically impossible. This is clear example of excess of freedom in our non linear grey box model. It was a little mistake that we realized at the end of the identification process and was impossible to rectify afterwards. The solution is to include in the m-file of the idnlgrey object a restrictive condition for the Static friction parameter ($\mu_S > \mu_C$).

Finally, the final value of offset voltage parameter (V_{off}) is 0.264V over its initial value but it is a necessary change due to the behaviour of Controllers block generating their output signal, as is shown in Figure II.40 this is not symmetrical about 5V and its mean is around 5.3V, thereby, the final value of V_{off} is consistent.

The model of STC inverted pendulum has been found. Next chapter is focused on representing this model using Matlab Simulink software.

3. MODEL REPRESENTATION ON SIMULINK

After finding an accurate model that represents faithfully our system the next step is to build it using Simulink blocks, thus obtaining a software model of the STC inverted pendulum. We begin this section showing separately the equivalent schematics of each of the blocks previously modelled, and it will conclude showing the final schematic, which includes all the model's blocks.

In this section will be explained every design decision taken, every problem appeared and the solutions adopted. Some explanatory diagrams and graphs will be shown to substantiate the proper functioning of the software model of the system.

Before starting the software model it is necessary to explain that every constant values, like the thirteen parameters values found out, used in this Simulink model are defined into a Matlab script that is executed at start of each simulation.

The main reason for doing a constant script is to separate the software model from its variable/constant values. If something changes in STC inverted pendulum system, for instance the pendulum mass *m*, there is nothing to change in the software model only it is necessary to write the new value of *m* in the constant script. It also will be useful in the future when some upsets will occur in our system (pieces wear, lubricants, etc.). Then we just have to make an adjustment of the current constants through the identification methods used previously and did not need to change anything in the software model

3.1. Simulink Software Model of Power Driver Controller, Motor AC, Timing Belt and Spindle

This software model using Simulink was really easy to do. As is shown in Figure III.1 we have translated the mathematical language (remember Equation II.1) into Simulink using; an input port Vin, which is the input voltage proceeding from Controllers block; a constant block with value equal to V_off, which is the offset voltage used in the Power Driver Controller; a subtract block, which does the voltage subtraction; and finally a gain block, which multiplies the voltage subtraction by alpha constant that is the voltage to force conversion factor.

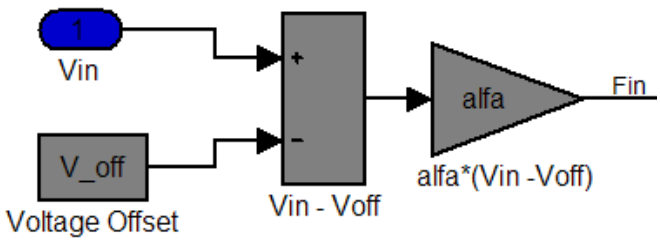


Figure III. 1: Simulink software model of Power Driver Controller, Motor, Timing Belt & Spindle block

Concluding this software model we want to recall that moments of inertia from Motor and Spindle are included in the constant block M (see Equation II.8).

3.2. Simulink Software Model of Pendulum block

This section is divided in two subsystems: Pendulum equations Simulink model, which has as inputs the force proceeding from Spindle (Fin) and the friction force produced in the cart without viscous friction ($f'_{FRIC\_X}(t)$); and friction force of cart Simulink model, which has as inputs the velocity of the cart (dx_dt), which is necessary to differentiate between static and dynamic friction force; and the force proceeding from Spindle (Fin), which is necessary to determine the static friction force (see Equation II.12).

3.2.1. SIMULINK SUBSYSTEM OF PENDULUM WITHOUT $f'_{FRIC\_X}(t)$

This model, which is shown in Figure III.2, represents with Simulink blocks the pendulum equations I.35 and I.36. But to reach this Simulink software model it is necessary answering some questions that we are going to explain. Remember that in this part of the total Simulink model the way to obtain $f'_{FRIC\_X}(t)$ (cart friction forces without viscous friction) does not matter, it is taken as input.

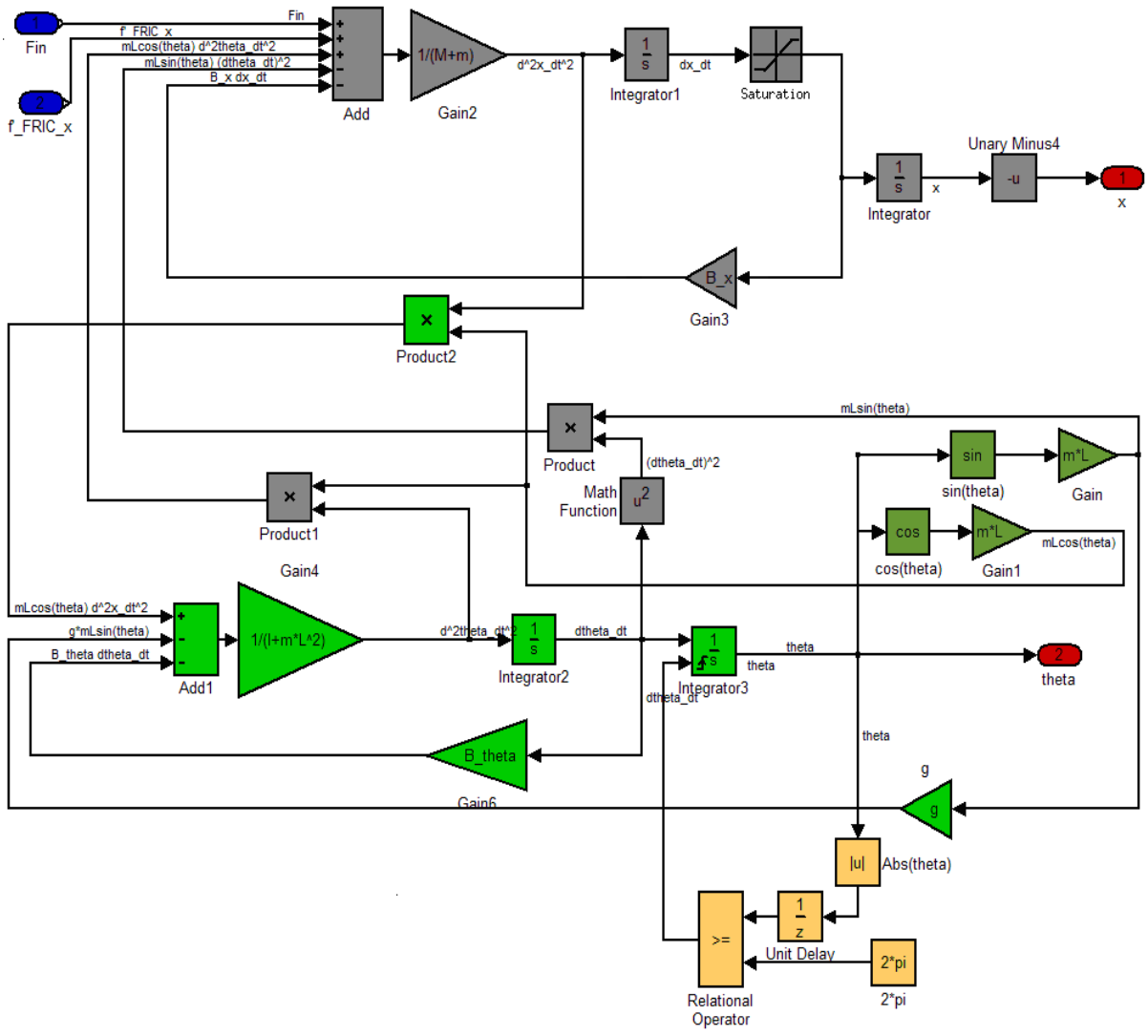


Figure III. 2: Simulink Subsystem of Pendulum without $f'_{FRIC\_X}(t)$

The first question that arose when we want to represent the pendulum equations, which are second order differential equations, was how to do it in Simulink?

The first idea is thinking in derivative blocks but they do not allow to store any initial conditions, and it could be an interesting idea for our model, for instance, starting the model with the pendulum in an inverted position (θ = π). Therefore, integrator blocks are the answer and we will need four integrators: two for each generalized coordinate (x and θ).

But how can we represent a differential equation in Simulink? First of all it is necessary to do some changes in Equation II.35 and II.36 because, as is explained in [22], we have to work out the highest order derivative terms, which are $\ddot{x}$ and $\ddot{\theta}$:

$$\ddot{x} = \frac{1}{(M+m)}\left(f(t) + f'_{FRIC\_X}(t) + mL\ddot{\theta}\cos\theta - mL\dot{\theta}^2 sen\theta - b_x\dot{x}\right) \quad \text{(III.1)}$$

$$\ddot{\theta} = \frac{1}{(mL^2+I)}\left(mL\ddot{x}\cos\theta - mgLsen\theta - b_\theta\dot{\theta}\right) \quad \text{(III.2)}$$

Analysing Equation III.2 for instance, there are three adding/subtracting terms, does not matter their values, multiplying for a constant gain and it is equal to $\ddot{\theta}$. Therefore, an adding/subtracting block and a gain block must be used and connected, the output from the gain block will be $\ddot{\theta}$. This wire output must be connected to an integrator block and its output, which will be the angular velocity $\dot{\theta}$, must be connected to the second integrator block to obtain the angular position of the pendulum θ.

Now we are ready to connect, doing a loop, the inputs of the adding/subtract block because we can take a data path from $\dot{\theta}$ or θ and, after using their respective sinusoidal and gain blocks to obtain the exactly value of these three terms, the differential equation in Simulink is done.

As is shown in Figure III.2 both differential equations are implemented using Simulink blocks:

- Grey blocks for Equation III.1.
- Green blocks for Equation III.2.
- Dark green blocks are used in both equations.
- Yellow blocks are used to keep θ between -2π and 2π.

Before concluding this section there are some Simulink blocks that need to be further explained:

Yellow blocks and $\dot{\theta}$ to θ integrator block

Sometimes the angle position θ is out of "one loop" boundaries (-2π, 2π), for instance, when the balance controller misses its target and the pendulum falls down. To ensure that the angular position values are always within the boundaries, it is necessary to include some logical blocks in our Simulink model. The objective is to replace the value of θ, when it is higher or equal than 2π or lower or equal than -2π, by zero.

First of all, the current value of θ is taken and its absolute value is calculated, connecting θ to an absolute value block; Next block is a delayer of one sample period, it is necessary to avoid algebraic loops problems, which occurs when a data path has two different values at same time (in our case -2π/2π and zero); then, a relational operation block of higher or equal compares this absolute value of θ with a constant block of 2π. To finalize, the output of this relational operation block is connected to a rising edge reset port in the $\dot{\theta}$ to θ integrator block because when the absolute value of theta exceeds 2π the output of relational operation block goes from 0 to 1 making a rising edge and resetting the integrator block to 0.

Unary Minus block

In Figure III.2, just before the red output port of x, there is a unary minus block. The reason of this change of sign in the cart position is the CKK definition of the positive direction to the left. As we define the origin of coordinates in plane XY at left side (see Figure II.9) the unitary vector that multiplies x is –i (see yellow box in Figure II.9), therefore, a unary block is needed.

Saturation block

In Figure III.2, just after the first grey integrator, which converts the cart acceleration into cart velocity, there is a saturation block. This block it is necessary because the maximum cart velocity is limited by the CKK module. Remembering Figure II.20, where is shown the permissible velocity for this Compact Module, the maximum velocity is 33 *m/min* that corresponds to 0.55 *m/s*. This limitation must be considered in our software model for the STC inverted pendulum, therefore a Simulink saturation block is placed on the cart velocity wire. Its upper limited is set to 0.55 and its lower limited is set to -0.55.

3.2.2. SIMULINK SUBSYSTEM OF $f'_{FRIC\_X}(t)$

This model can be divided in two parts: friction force with cart movement or Coulomb friction force, when the velocity of the cart is different to zero; and friction force without cart movement or Static friction force, when the velocity of the cart is equal to zero. The value of $f'_{FRIC\_X}(t)$ will be the output of a switch block that has as inputs both frictions and as selector the absolute value of velocity of cart that is used as criteria for switching depending on if this value is higher than zero or not.

Friction force with movement

In Figure III.3 it is implemented the Equation II.106 using Simulink blocks. Such as this friction force depends on the sign of the cart velocity, it is an input in Figure III.3. This Simulink model is very easy to do, the only thing to emphasize is the Math Function block of Simulink that is used as Pow block and Exp block in our model. After dragging this Math Function block in our schematic and double clicking, it is possible to select the type of function (Pow, Exp, etc.)

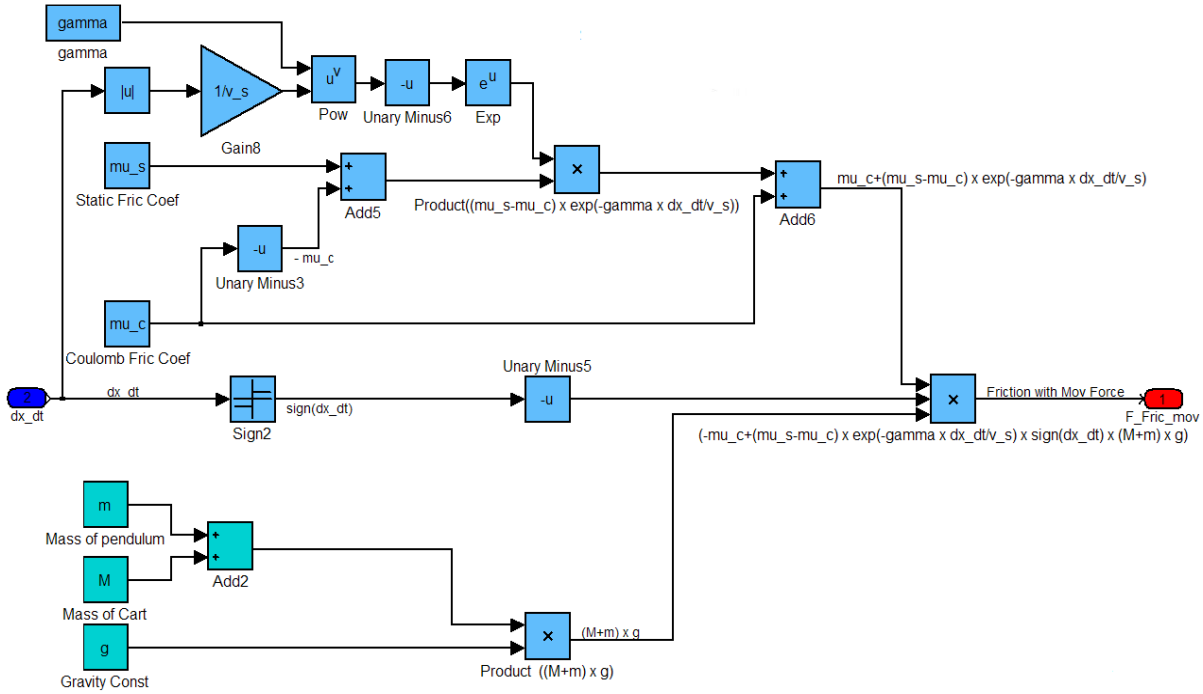


Figure III. 3: Simulink software model of friction force with cart movement

Friction force without movement

The Equation II.12 describes Static friction force and in Figure III.4 is shown this equation using Simulink blocks. The absolute value of input force, which is named *f(t)* in Equation II.12 and Fin in Figure III.4 (see dark blue input port), is the variable for which Static friction force depends. The switch is used to select the right value of Static friction force, as criteria for passing first input is $\left|f(t)\right| \geq \mu_{S\_X} F_N$, where $F_N = (M+m) \cdot g$.

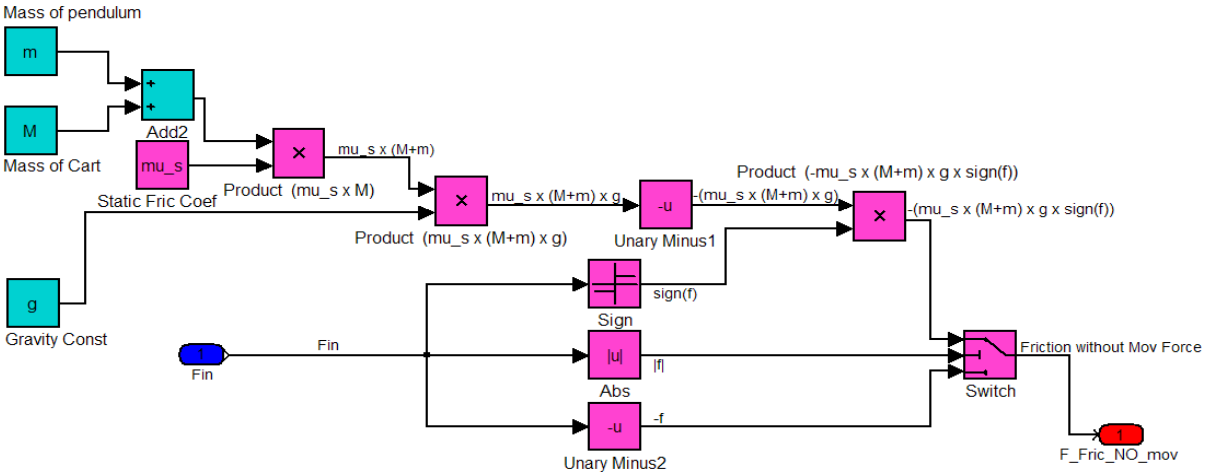


Figure III. 4: Simulink software model of friction force without cart movement

For concluding this subsystem it is necessary to show how $f'_{FRIC\_X}(t)$ is calculated depending on cart velocity. Friction force with movement or without it are selected using a Switch block, which criteria for passing first input is that the absolute value of cart velocity will be higher than zero, as is shown in Figure III.5.

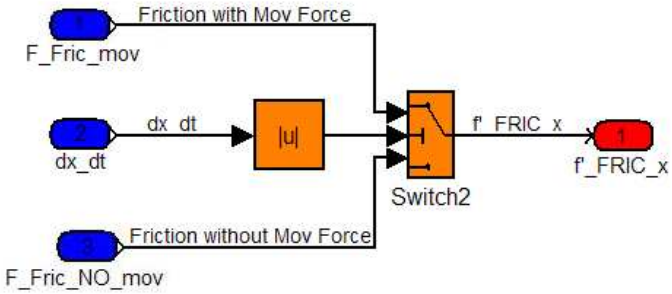


Figure III. 5: Simulink Subsystem of $f'_{FRIC\_X}(t)$

Finally, the Simulink software model of pendulum block is shown in Figure III.6:

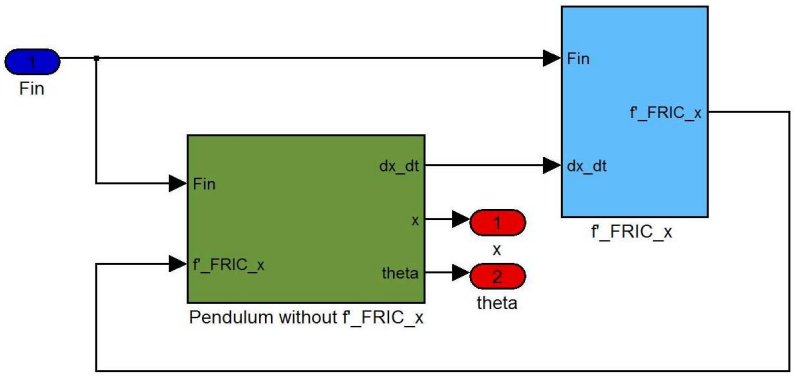


Figure III. 6: Simulink Software Model of Pendulum Block: Input, *Fin*; Outputs, *x* and *θ*.

3.3. Simulink Software Model of Encoders block

The last block, in STC inverted pendulum software model, is the Encoders block. There are two incremental rotary encoders to model: cart position (x) and angular position (θ) encoder. Therefore, this section could be divided in two parts, one for each encoder, but it is not divided because their Simulink schematics are really similar. Thereby, the cart position encoder will be taken as example to explain how the Simulink software model of encoder has been designed. For understanding the angular position encoder we only need to know their differences.

The unique differences between both encoders are:

- The input data: cart position (x) or angular position (θ).

- An extra gain block in cart position encoder: that is necessary for converting the increment of cart position (x) into increment of angular position of the Spindle (φ) as is shown in Equation II.55.

- The values of angle threshold: that defines the width of a disk band, which is different in each encoder due to their different resolutions (see Equation II.49 and II.50).

3.3.1. SIMULINK SCHEMATIC OF CART POSITION ENCODER

In Figure III.7 is shown two Simulink schematics of the cart position encoder model. At left side there is the top hierarchical level of this software model, where a Matlab vector with the cart position data is the input and the train logic pulses from both channels (A and B) are the outputs; at right side the second hierarchical level of this Simulink model is shown, we can see the subsystems corresponding to both channels.

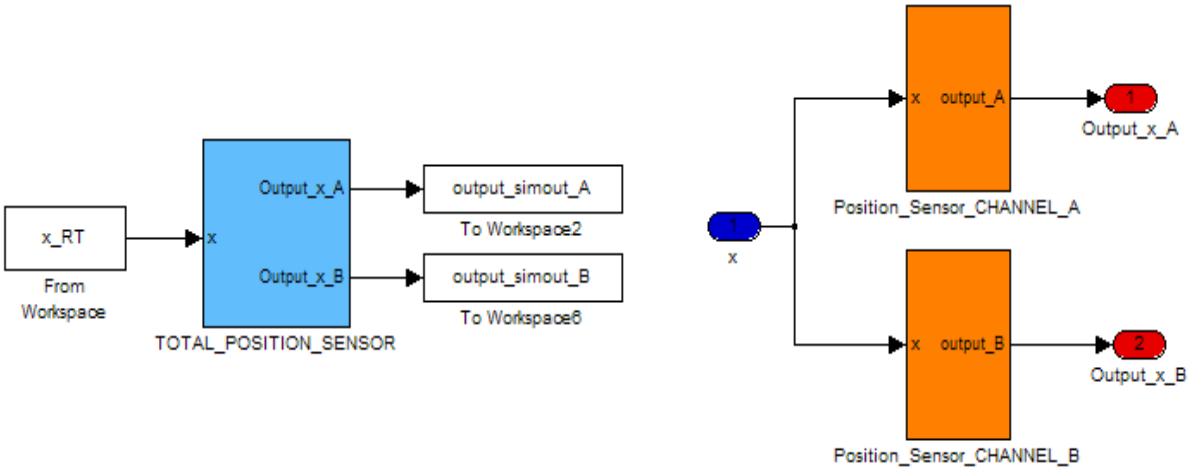


Figure III. 7: Simulink schematics of cart position encoder. Top (left) & 2nd (right) hierarchical levels

The basic ideas have been explained in the second chapter, section 2.4. Modelling Encoders, but this section is focused on Simulink implementation. As is shown in Figure III.8, the red input *x* (the cart position) is connected to the cyan block Incrementer x, where the increment of cart position (Δx) is obtained as is shown in Figure III.9. This Incrementer block works subtracting the last value of x, which is stored in a memory block, to the current value of x.

Next block in Figure III.8 is a gain block that is used to obtain the increment of angle (Δφ) in the encoder disk, remember Equation II.55. This block does not exist in angular position encoder because there the input is θ, therefore, the Incrementer block gives directly the increment of angle (Δθ) in the angular position encoder disk.

This increment of angle is used to calculate the current angle, which is the FSM input. As has been explained in the section 2.4., the current angle is obtained as the addition of its last value plus the increment of angle (see Equation II.60) and this is the function of the addition Simulink block shown in Figure III.8 that is placed just before FSM chart block (dark green block in Figure III.8).

Next block to explain is the magenta switch in Figure III.8. First of all, forgetting for a while the memory blocks (grey blocks in Figure III.8), the four switch inputs corresponding with four FSM states: Input 0 → Darkness; Input 1 → Lightness; Input 2 → Lower_lower_limit; and Input 3 → Over_upper_limit. Obviously, the select input corresponds to the current state. Returning to the memory blocks are necessaries because without them a trouble appears when Matlab tries to solve the algebraic loop that is in our Simulink model. Note that an algebraic loop in Simulink occurs when an input port with direct feedthrough is driven by the output of the same block, either directly, or, as in our case, by a feedback path through other blocks with direct feedthrough (see Figure III.8). The problem is solved adding memory blocks that represent a delay of one period.

As seen in Figure III.8 the switch input ports 0 and 1 are directly connected to current angle, or rather, to last angle (see the previously memory block). But the switch input ports 2 and 3 are connected, forgetting memory block, to an Add and Subtract block respectively. The reason has been explained in the section 2.4., and an explanatory diagram has been shown in Figure II.14. Such as we works with only two bands of the encoder disk when the angle is out of these boundaries an adjustment is necessary: adding 2**angle_th* when the angle is lower than the lower limit (state Lower_lower_limit) or subtracting 2**angle_th* when the angle is higher than the upper limit (state Over_upper_limit). This switch output is the last angle "adjusted" that will be added to the increment of angle to obtain the current angle.

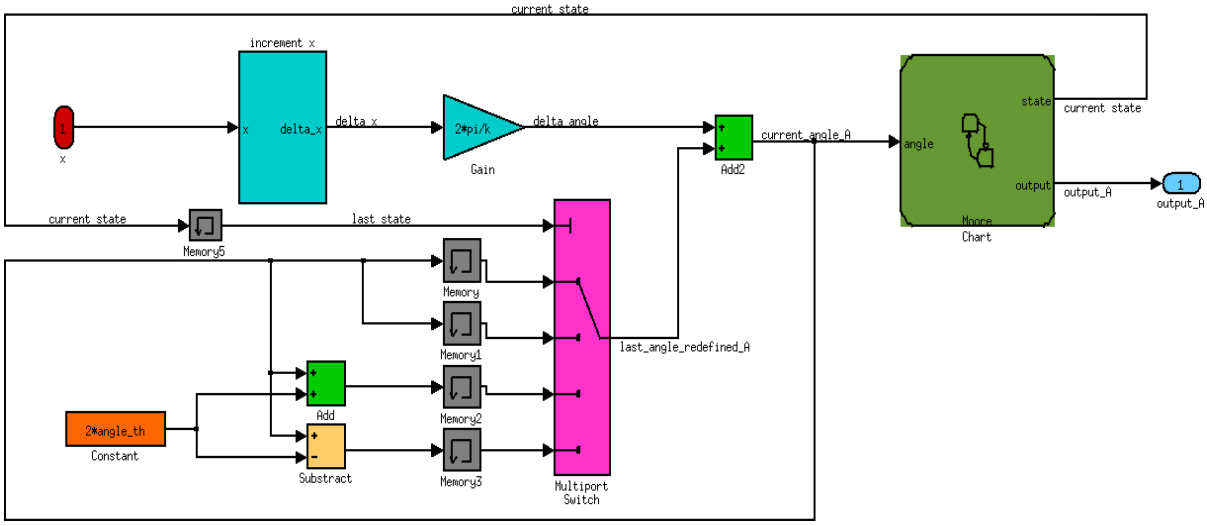


Figure III. 8: Simulink schematic of channel A for cart position encoder

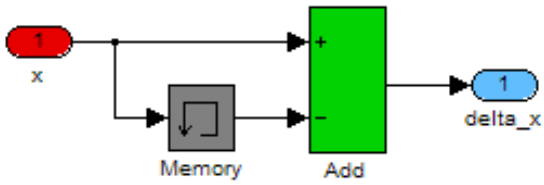


Figure III. 9: Incrementer schematic used in Simulink model of cart position encoder based on Equation II.60.

The Simulink schematic of channel B for cart position encoder is shown in Figure III.10. Practically it is equal to channel A, but there are some differences that must be explained. Due to a half band distance between sets of LED/photodiode that is necessary to obtain outputs in quadrature (cf. section 2.4.) following changes have been done in channel B model:

- **An extra Subtract block**: it is necessary for subtracting a half band, which is represented by a constant block with value of *angle_th*/2, to the current angle for adapting it to FSM channel B block input.

- **StateFlow Chart different transition conditions**: StateFlow diagram for channel B has been shown in the second chapter (Figure II.16) and it is equal to channel A if we replace the transition conditions of *-0.5* threshold* by *–threshold* and *1.5* threshold* by *threshold*.

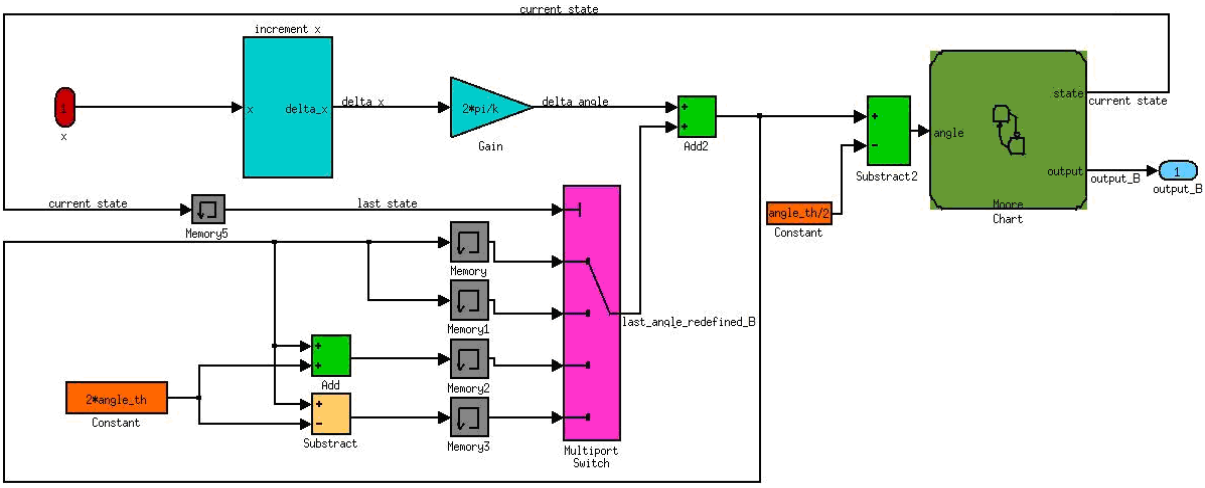


Figure III. 10: Simulink schematic of channel B for cart position encoder

Such as reminder, the Simulink schematic of angular position encoder is exactly the same but changing: the input x by θ; the gain block of $2\pi/k$ by nothing; and the constant value of *angle_th* by *0.2π mrad*.

3.4. Results for Simulink Software Model of STC Inverted Pendulum

In this section we are going to show the results obtained using our Simulink software model of STC inverted pendulum comparing it with the real system. These results are divided in two parts: Simulink software model of Power Driver Controller, Motor AC, Timing Belt, Spindle and Pendulum block; and Simulink software model of Encoders.

3.4.1. RESULTS OF SIMULINK SOFTWARE MODEL OF POWER DRIVER CONTROLLER, MOTOR AC, TIMING BELT, SPINDLE AND PENDULUM BLOCK

Setting the simulation parameter of our Simulink software model using the variable step Adams-Bashforth-Moulton type as solver because it was used previously in identification parameter section (cf. the properties of idnlgrey_object in section 2.5.4). Setting 10^{-3} and 10^{-6} as values for relative and absolute tolerance fields respectively in this configuration parameters dialog every variable is exactly the same that into the Matlab non-linear greybox object used for identifying our model. Therefore is expected the same results for our Simulink software model that Figure II.40 for the firstly 7 seconds and quite similar results that Figure II.39 for the whole simulation time.

Unfortunately the results are different than expected, as is shown in Figure III.11. These differences can be because of the ODE solver behaviour into Simulink and the PEM method. The ODE solver of PEM method calculates next state vector, which is an input of our model file ("*inv_pendulum_m*"), using the state derivative vector, which is an output of our model Matlab file, and this process is different to ODE solver used in Simulink, where there is no state vector, instead of that there are Simulink blocks connected to continuous-time integrators. Both models should be equivalents but, as is shown in Figure III.11, they are not. It is possible that the zero crossing problem of our cart velocity signal could affect to integration process, too.

As is shown in the right side of Figure III.12, there are a high number of zero-crossing at the cart velocity signal in our Simulink software model and this generates an error because there is a zero-crossing control for variable step solvers. It is necessary to disable this zero-crossing control in the configuration parameters dialog for running without errors our Simulink software model. Remember that cart velocity signal is the key parameter for selecting between static and dynamic friction force and these zero crossing produce many swapping between them.

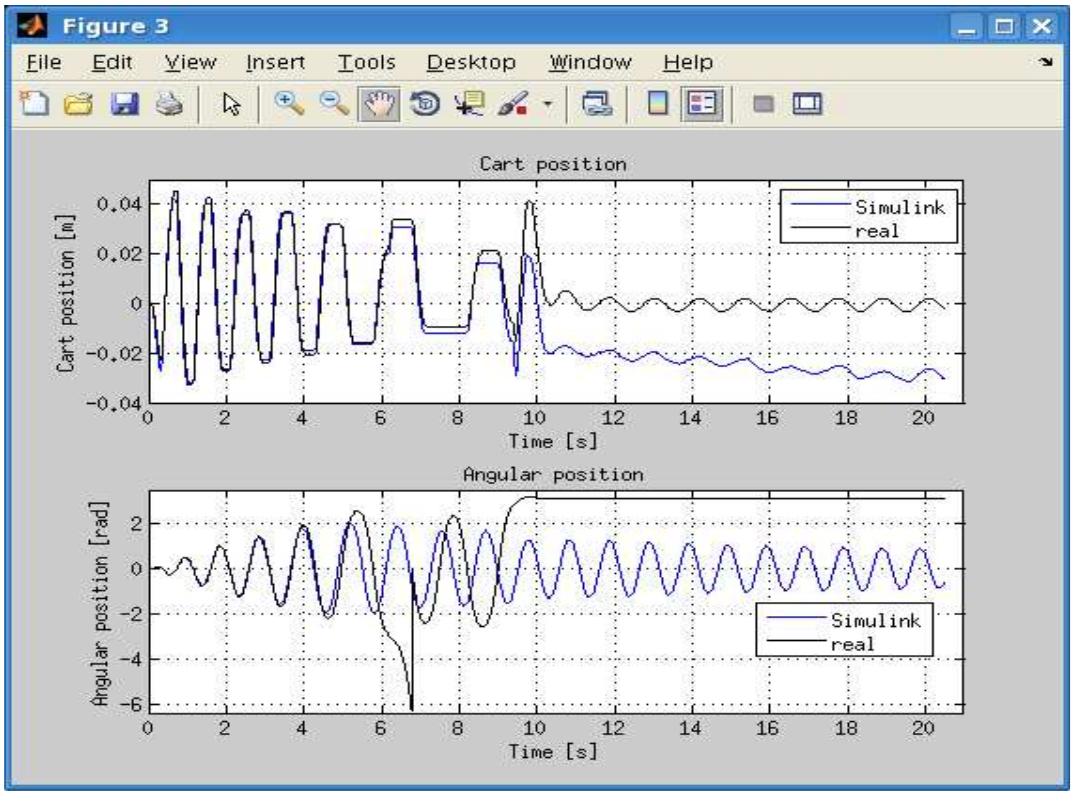


Figure III. 11: Comparison figure of software model and real pendulum outputs

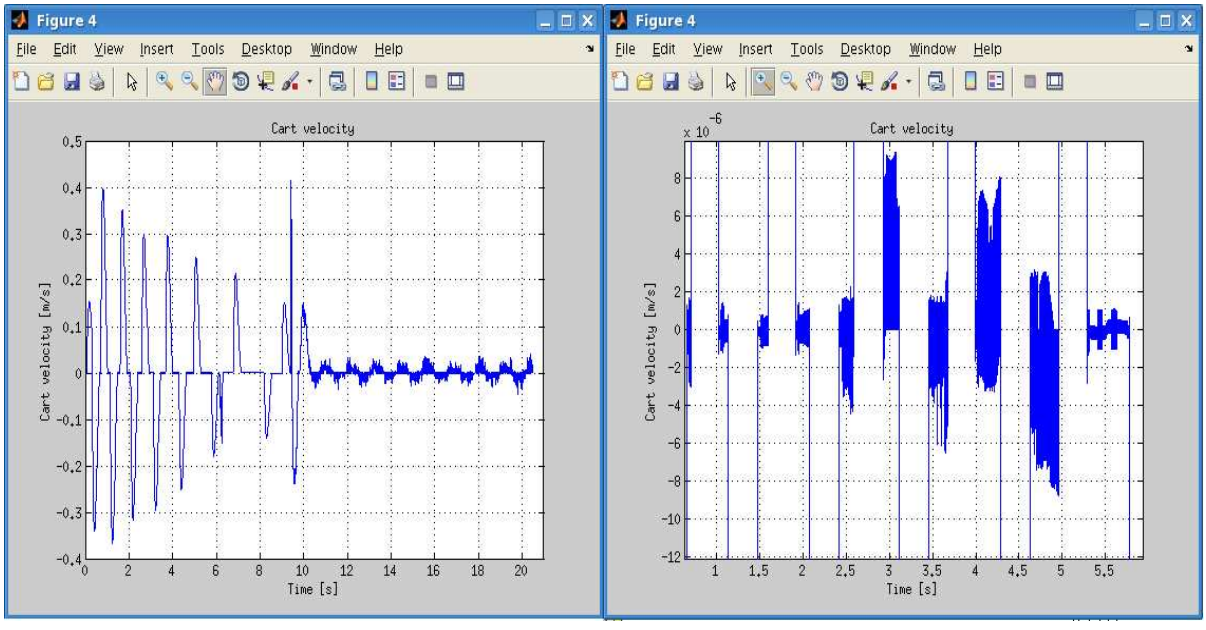


Figure III. 12: Cart velocity of Simulink software model (left) and its zoom (right)

To solve the zero-crossing problem it is possible to change the velocity cart threshold value of switching between friction forces. In theory this value is zero (see Equation II.106), but it is possible to consider cart velocities lower than one centimetre per second as zero. Therefore, changing this threshold value the zero-crossing problem is solved but the model has been changed, for instance the velocity cart is different as is shown in Figure III.13.

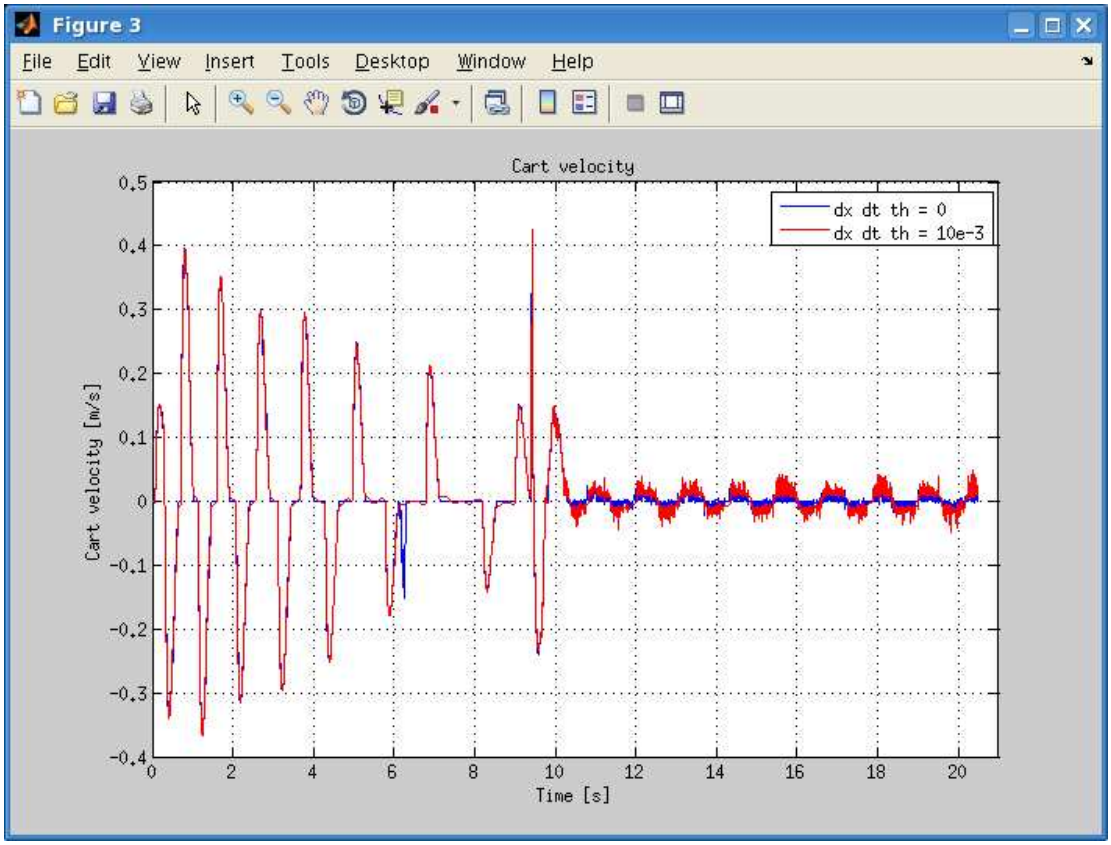


Figure III. 13: Cart velocity signal with threshold equal to zero (blue) and equal to 10^{-2} m/s

An alternative solution for zero-crossing problem is to use a fixed step solver, for instance ODE 1 that uses Euler's method as integration technique, because this type of solver does not need zero-crossing control due to its accuracy and time duration depends on the size of the steps taken by the simulation. In this case the fixed-step size selected has been $1\mu s$ instead of default value ($0.1ms$) because the accuracy is inversely proportional to step size and we need high accuracy.

This Simulink software model results are practically equal to variable step model as is shown in Figure III.14. However, the differences between our Simulink software model and real pendulum outputs are still quite significative, especially angular position. The problem lies in switching controller response because firstly six second the model is pretty fit and after this time, when it is produced the switch to Balance controller, the cart position is mismatched and it goes worse at 9.5^{th} second, when Balance controller takes the control.

Analyzing angular position, it seems that every time the pendulum swings the angular position needs just a little bit more of amplitude. That is the reason why the mismatch between our model and actual pendulum is increasing until 6^{th} second. After this time, when the real angular position makes a loop, both signals are completely different but it is consistent with input voltage, for instance after 9.5 second input signal is given by Balance controller thereby it is impossible that angular position reaches the inverted equilibrium.

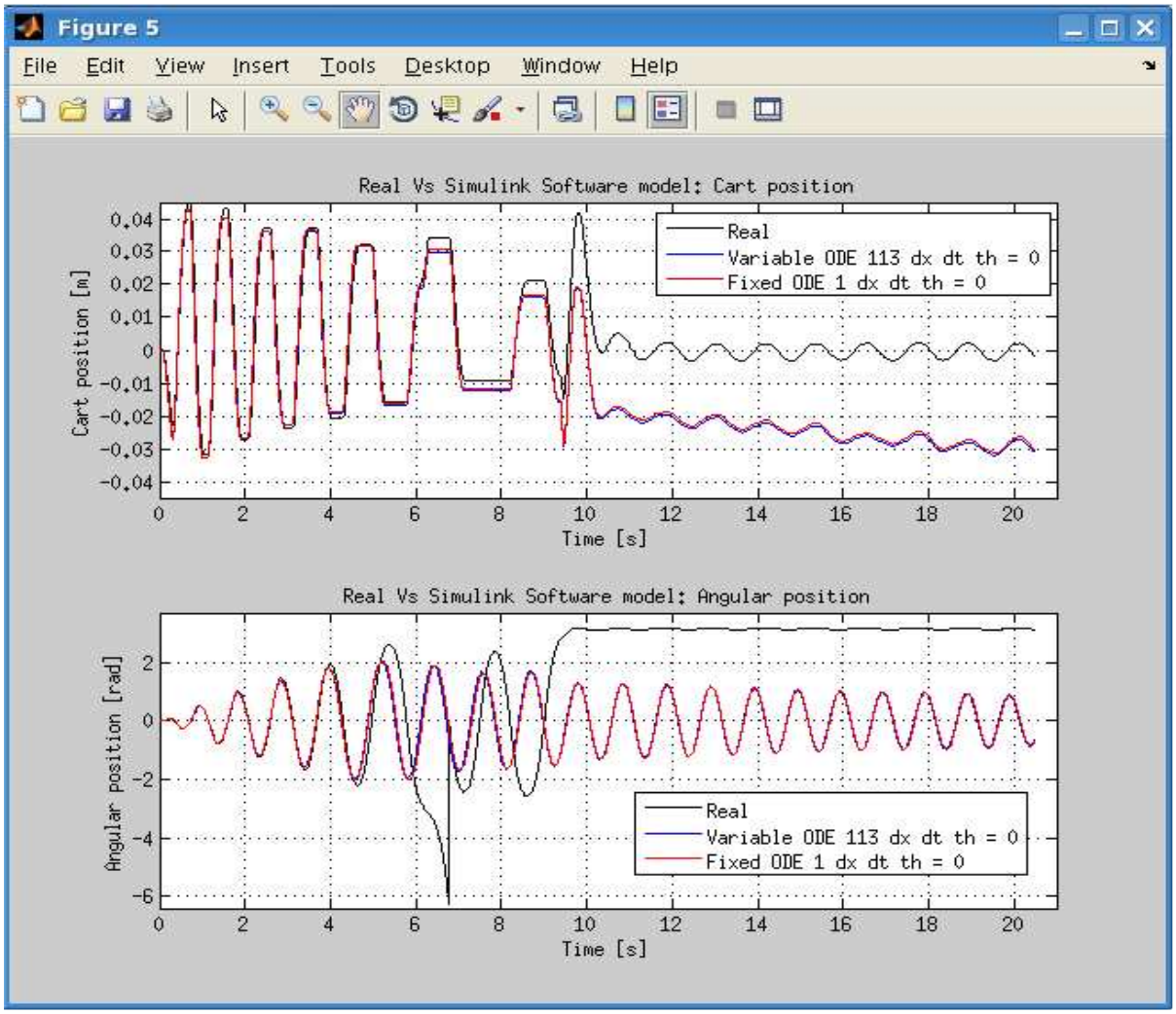


Figure III. 14: Comparison figure of both software models and real pendulum outputs

While results have been somewhat disappointing, especially after the great amount of time invested in identifying the model parameters, we think that we are not far from the fitted model and possibly the controllers block is capable of balance the pendulum in the inverted position. Therefore, we have to see what results we get from the hardware model in order to conclude something.

Next step is to analyze the results obtained for our Simulink software model of Encoders.

3.4.2. RESULTS OF SIMULINK SOFTWARE MODEL OF ENCODERS

In this section the results obtained after testing these Simulink software models are shown. There are two different ways for testing our encoder models:

- Doing an experiment with STC inverted pendulum system and measuring outputs of real encoders, storing these trains of pulses signals and also the cart and angular position data. After this experiment we can introduce as inputs in our models the cart an angular position data stored previously, obtaining our encoder outputs and comparing them with the real encoder outputs.

- Introducing some training data of cart and angular position in our encoder models, and using their train pulses outputs as inputs in the Simulink schematic that is included in Controllers block for converting the real encoder outputs signals into cart and angular position, and comparing these final positions with the initials.

Either option could have been selected to test our encoders, since the two are completely valid and sufficient by themselves. But finally, the second option was selected because it was easier than the first one. Note that for measuring and storing the outputs of real encoders some new design in Controllers block is necessary, on the contrary, the second option does not need any extra design in Controllers block. Furthermore, remember that Controllers block is the only block in SCT inverted pendulum system that does not belong to this master thesis.

Thereby, the data training used as entries in encoder models were selected from a real experiment of SCT inverted pendulum system, as are shown in Figure III.15. This graph shows how Controllers block works: the first six seconds, it is working the swing up controller, which moves the cart like a swing forcing the pendulum to increase its angle; at sixth second the balance controller starts to work, but unfortunately it can not keep the pendulum in inverted position (-π rad in this case) and the pendulum falls down, then the swing up controller is activated again; finally at tenth second, the balance controller returning to work, and this time it can balance the pendulum until the end of the experiment (π rad in this occasion).

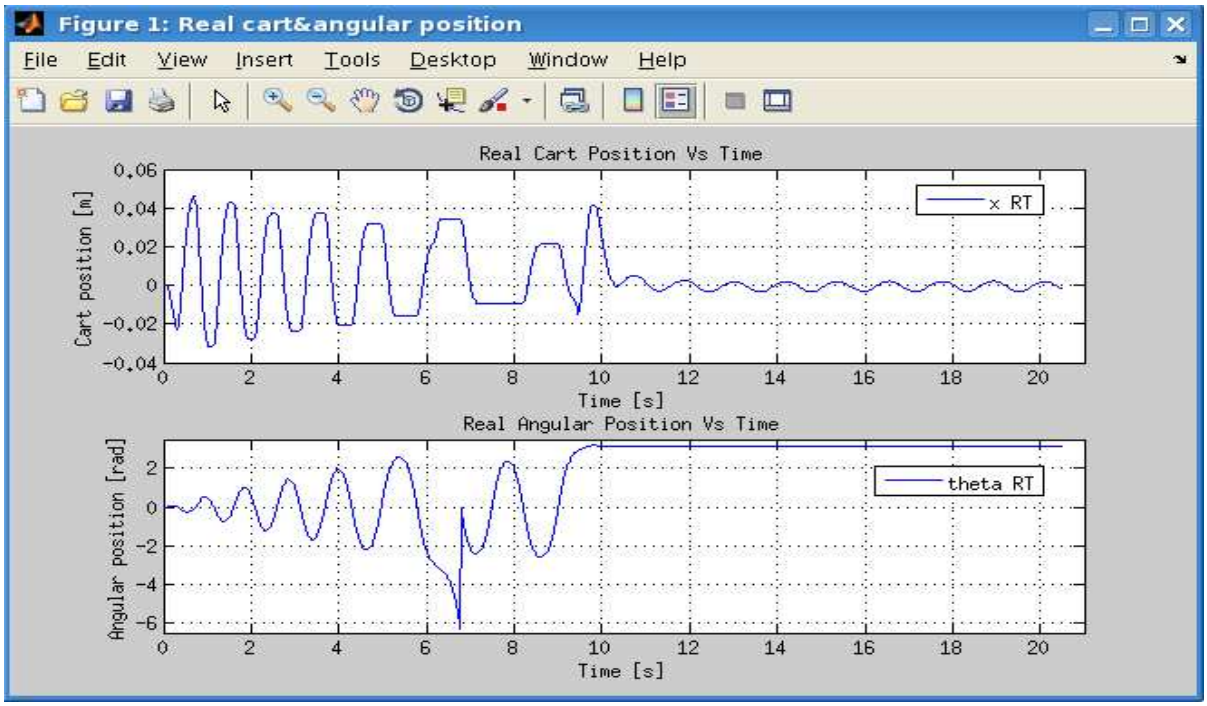


Figure III. 15: Cart and angular position stored in a real experiment of SCT pendulum

After executing our encoder models, their outputs, which are trains of pulses and have been stored in a Matlab vector at Workspace, are used as inputs in the part of Controllers block that converts the real encoder signals into cart and angular position (see red blocks for θ and green blocks for x in Figure III.12). This Simulink schematic, which is part of the hardware model of Controllers, is shown in Figure III.16 and as seen there are Xilinx blocks that will be explained in next chapter.

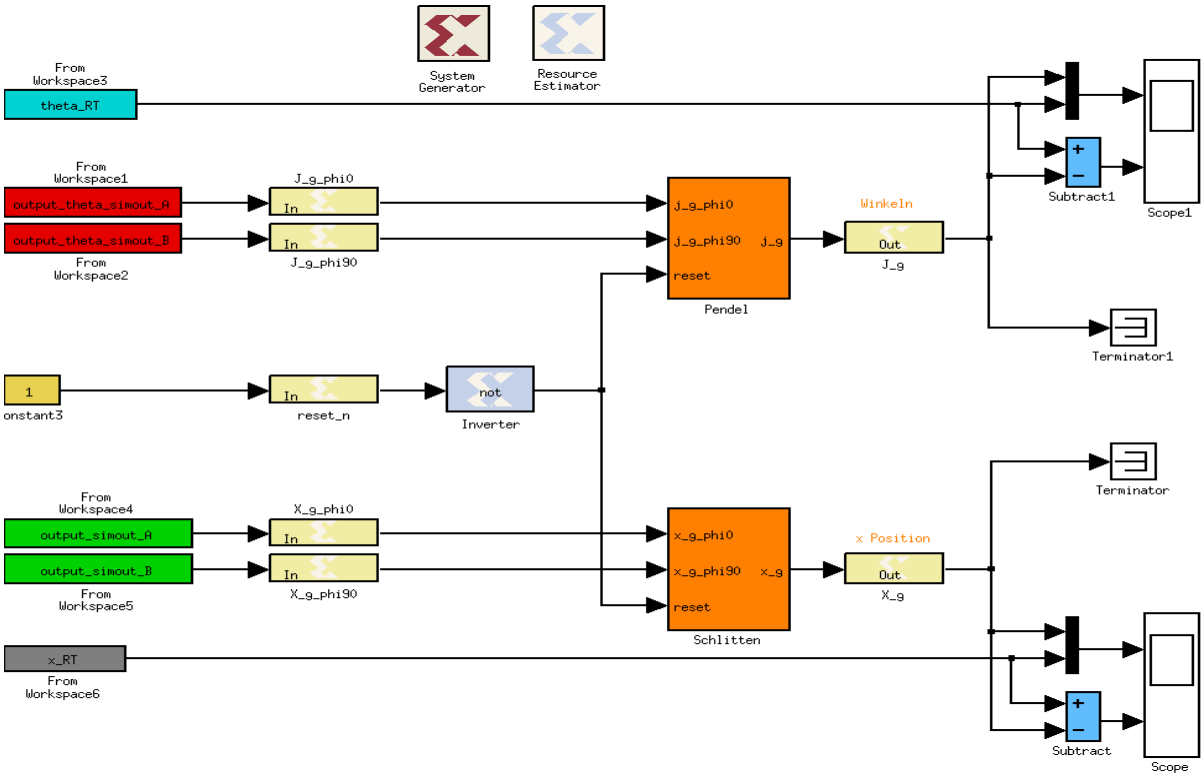


Figure III. 16: Simulink hardware schematic that converts train of pulses into cart/angular position

This schematic basically works counting edges from the train pulses inputs and with this information it can convert theses pulses into cart/angular position (outputs from orange blocks in Figure III.16). It is not necessary a deeply explanation because it does not belong to this master thesis, we only need to know that this works with the real system and must works also with our model. Therefore, if it does not work, the problem will be in our model design and not in this schematic.

For checking our encoder models two Simulink scopes have been placed in this schematic, see Figure III.16, the first input of these scope blocks is a multiplexer block that mixes real cart/angular position, which are vectors in our Matlab Workspace (see grey and cyan block in Figure III.16), with our model outputs that are named X_g and J_g for cart and angular position. In Figure III.17 are shown real angular position (Magenta line) and our angular position (Yellow line), and a first sight they are rather similar.

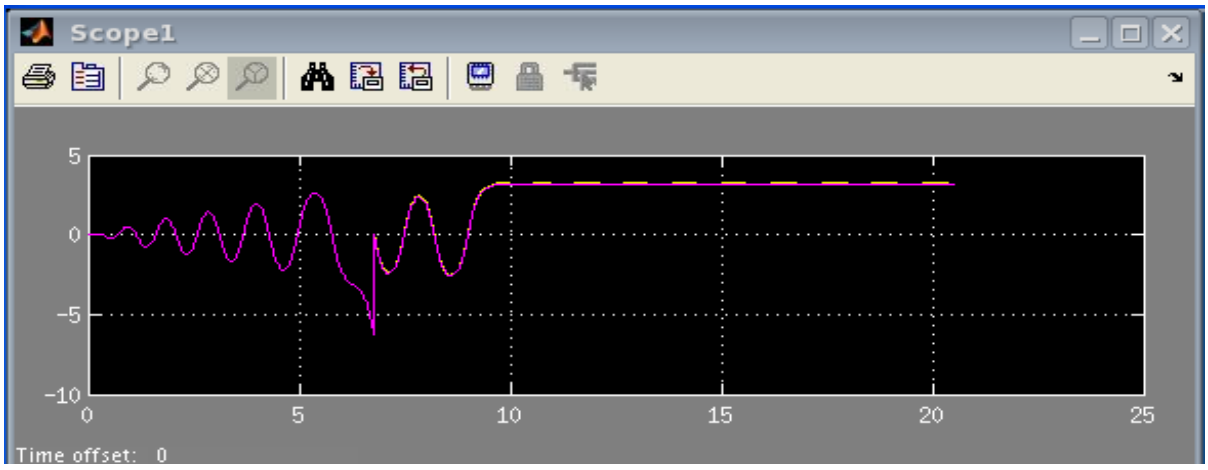


Figure III. 17: Real angular position (Magenta) Vs angular position after encoder model (Yellow)

In Figure III.18 are shown real cart position (magenta line) and our cart position (yellow line). This time the similarity between both signals is even greater, practically, yellow line is not seen.

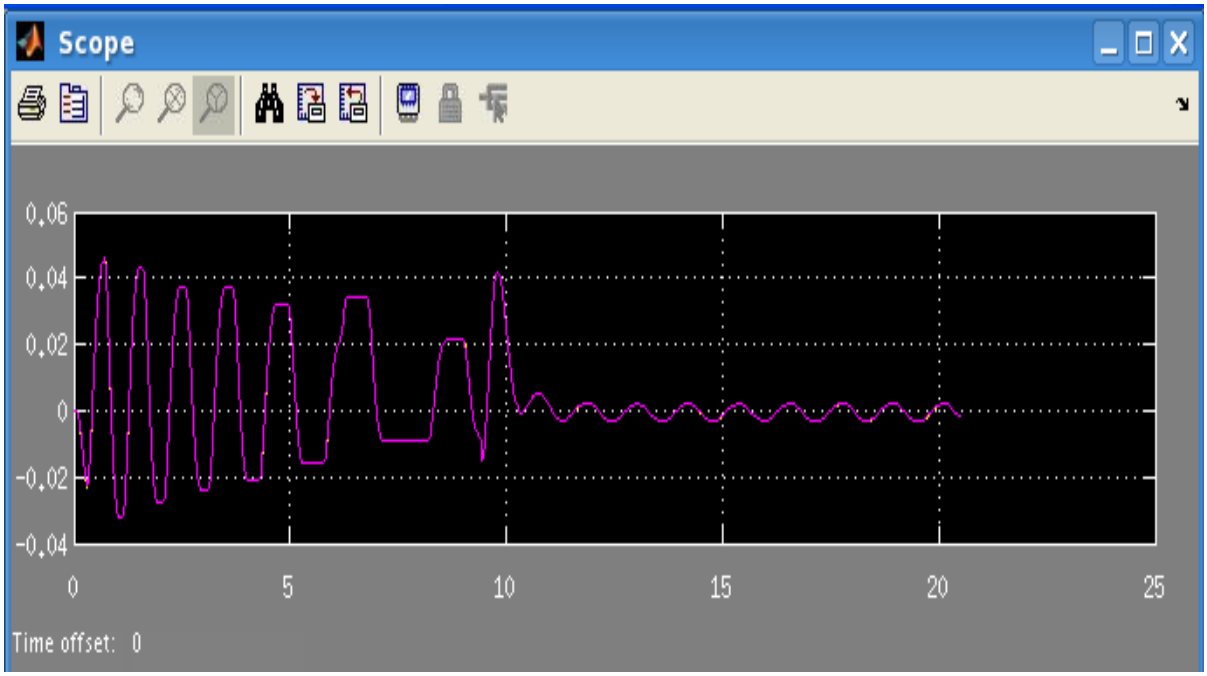


Figure III. 18: Real cart position (Magenta line) Vs cart position after encoder model (Yellow line)

These results are quite good, but we need another type of graphic that we can better appreciate the differences and thus verify the proper operation of our sensor model. For this reason, there are two subtract blocks in Figure III.16, which are placed just before the scope blocks, and that are used to obtain the difference between real and modelled signals. See Figure III.19 and Figure III.20.

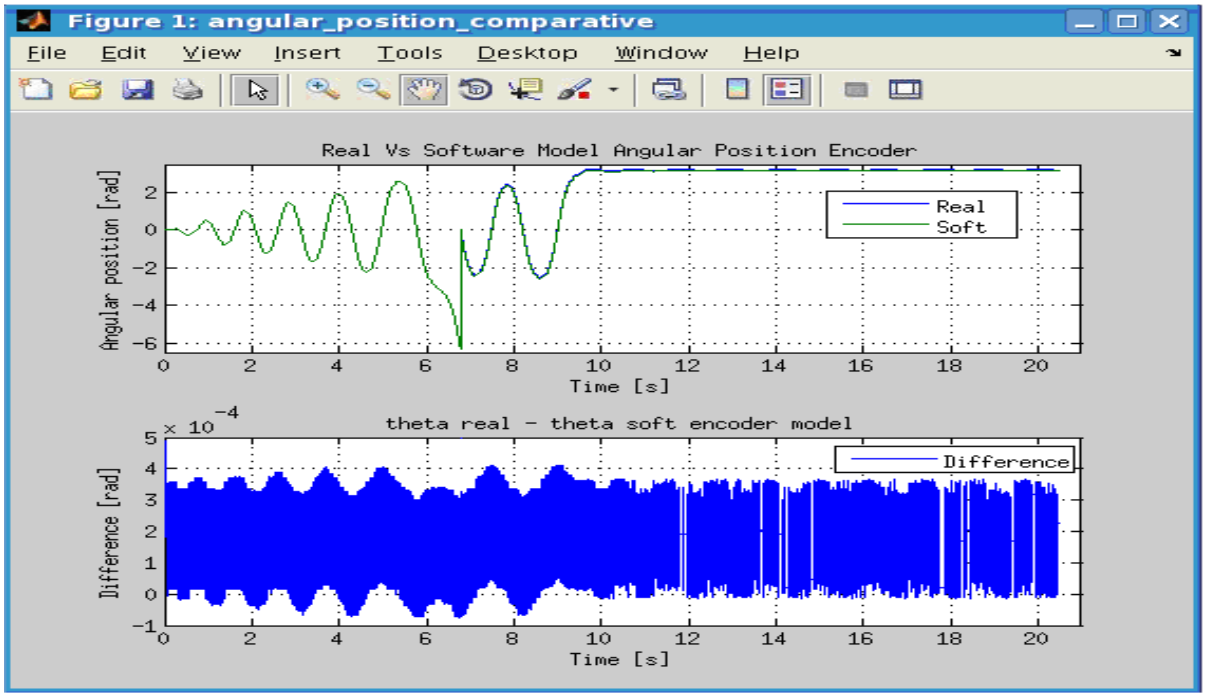


Figure III. 19: Real Vs Software angular position encoder and their difference.

As is shown in Figure III.19 the difference between Real and Software that gives an idea about the error of our software model encode. We can assert this software model works perfectly because the maximum error is *0.4 mrad* and it is very small, one and a half times lower than the band width of angular encoder disk (remember Equation II.50).

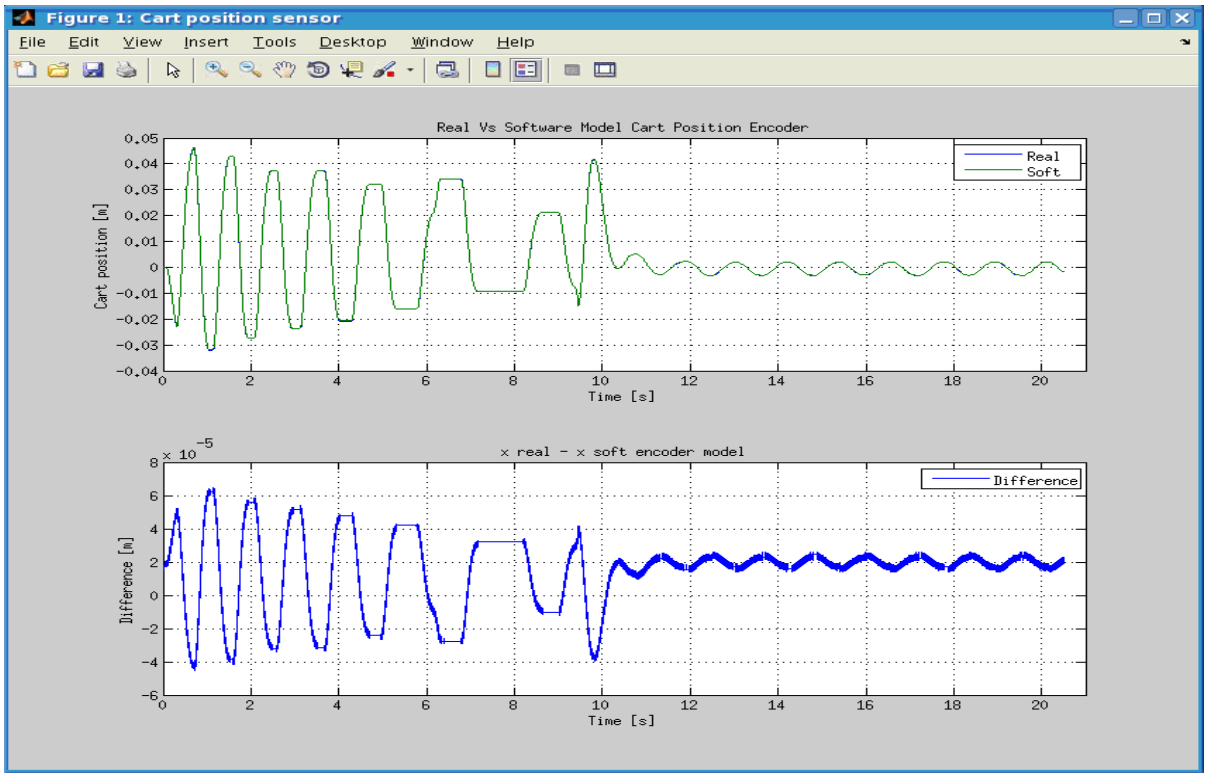


Figure III. 20: Real Vs Software cart position encoder and their difference.

As is demonstrated in Figure III.20 our Simulink software model for cart position encoder is very accurate, even more than angular position encoder, because the maximum difference between Real and Software signal is less than *70 μm*. As seen, this error is negatively proportional to cart position value, but it is not a problem because the maximum cart position value is *0.5 m* (ten times more that in this experiment) therefore the maximum error will be *0.7 mm* for *0.5 m* and it still is not a problem. The explanation for this behaviour of the difference signal (blue line in Figure III.20) for the cart position encoder is due to a very short delay in the output signal of our encoder model that produces this negatively proportional behaviour of the difference signal, when the subtraction between real and software model cart position signal is done.

However, these models of encoders are very accurate and work very well. We know that probably there is an easiest way to model the encoders (for example calculating the increment produced in the input signal (cart or angular position) in a fixed time period (i.e. ten sample periods) and then building the output signal with the number of pulses that corresponds to this increment). But modelling them as we have done it is better if we think in real time, because their maximum delay is equal to a sample period ($Ts = 1\mu s$).

This way to model describes the real encoder system exactly as it actually works, and that is always very important when it comes to modelling. Another interesting example of an incremental rotary encoder model using Simulink is reported by J. J. Incze et al. [23], they do not use a FSM to generate the encoder outputs instead of it they use logical circuits, relational operators and two T-type (Toggle) flip-flops as is shown in Figure III.21.

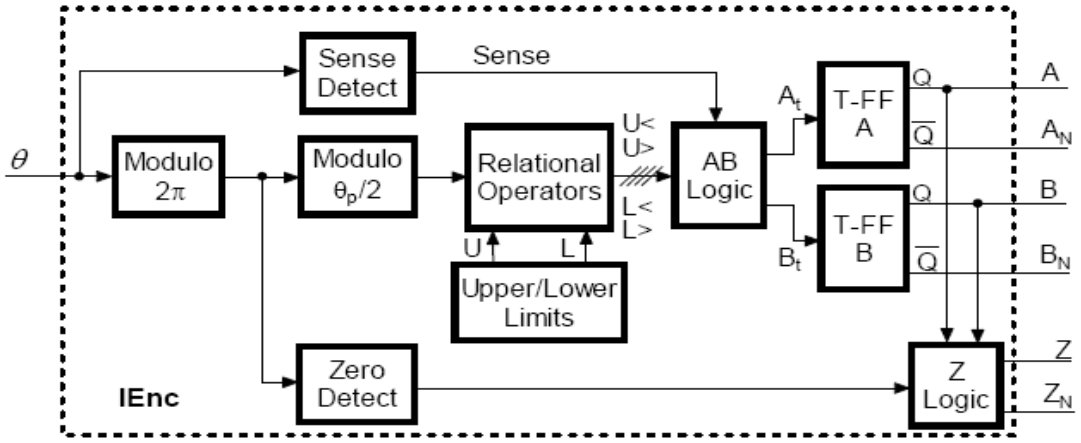


Figure III. 21: Simulation structure of the alternative encoder model (Figure 3 of [23])

The first point to note in this different model is that it has a third output signal named Z, which is used by some incremental rotary encoders in order to have a reference position. It is achieved equipping the encoder, as is shown in Figure III.22, with a third set of LED/photosensor and a second track on its disk having a single transparent band that produces on the output Z a single pulse when the encoder shaft completes a rotation (±2π radians). In our case, neither of our two encoders have this third output, therefore we will focus on the output signals A and B.

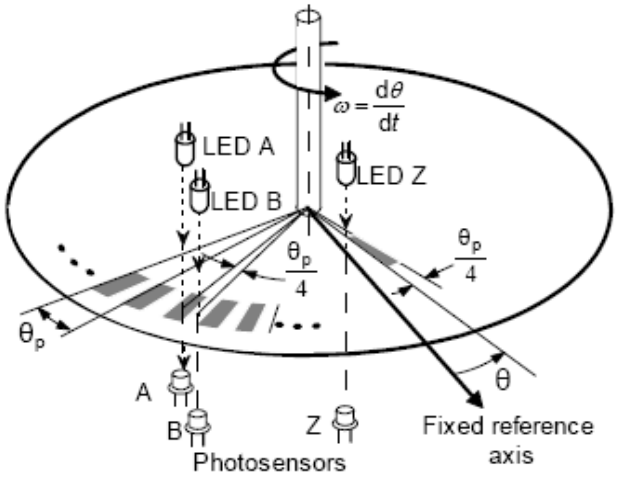


Figure III. 22: Construction principle of the alternative encoder model (Figure 1 of [23])

Next point to emphasize is the angle named as θ_P (i.e. angular step of the encoder) in Figure III.22 that corresponds to two times our *angle_th* (c.f. Figure II.14) because it spans two bands (transparent and opaque) of the encoder disk and it is defined as is shown in Equation III.3.

$\theta_P = 2\pi / N_R$ (III.3), with N_R as encoder resolution (number of angular steps on the encoder disk)

This angle value is basic for describing the encoder model outputs in a different way to us: using *modulo* operation, which returns the remainder of a division, as is shown in Equation III.3 and III.5.

$$A(\theta) = \begin{cases} 1 \leftrightarrow 0 < \mathrm{mod}_{\theta_P}(\theta) < \theta_P / 2 \\ 0 \leftrightarrow \theta_P / 2 < \mathrm{mod}_{\theta_P}(\theta) < \theta_P \end{cases} \text{(III.4)} \quad B(\theta) = \begin{cases} 1 \leftrightarrow 0 < \mathrm{mod}_{\theta_P}(\theta - \theta_P / 4) < \theta_P / 2 \\ 0 \leftrightarrow \theta_P / 2 < \mathrm{mod}_{\theta_P}(\theta - \theta_P / 4) < \theta_P \end{cases} \text{(III.5)}$$

This idea consists on applying the modulo operation to the angular position of the encoder shaft (θ) using as divisor the angle θ_P, doing this we are sure that the input angle (θ or θ - $\theta_P/4$ depending on the channel output that is being modelled) is always inside the boundaries of the operation area of the encoder model (c.f. Figure II.14), which values are 0 and θ_P in this case.

This is a brilliant and easy way for modelling the output channels of the encoder, but this idea was dismissed by us because it implies to divide, and, as it is well known, this is the most complex of the four basic arithmetic operations for the hardware, and it is really hard if the dividend (i.e. input angle) and/or the divisor (i.e. θ_P) are rational or irrational numbers (as is our case, remember the θ_P value).

Instead of using modulo operation, in our software model of the encoders we use addition or subtraction for keeping the input angle always inside the operation area of the encoder model, as was explained in section 2.4. Modelling Encoders. This idea allows us to design a hardware model of the encoders starting from the software model as will be shown in the next chapter.

Thereby, this approach reported by J. J. Incze et al. [23] is a perfect software model of a rotary incremental encoder that can be easily represented on Simulink but it is not very useful as hardware model.

To conclude this section we can compare the results obtained by our software model of the encoders and the results obtained by the approach given by J. J. Incze et al. [23], which are shown in Figure III.23. They are clearly worse than those obtained by our encoder model (c.f. Figure III.19). Quantitatively, this encoder model is five times worse than our own model because the maximum difference between the real angular position (i.e. Theta $_{ref}$ in Figure III.23) and the computed angular position (i.e. Theta in Figure III.23) is around 2 *mrad* and the maximum difference in our encoder model is around 0.4 *mrad* as is shown in Figure III.19.

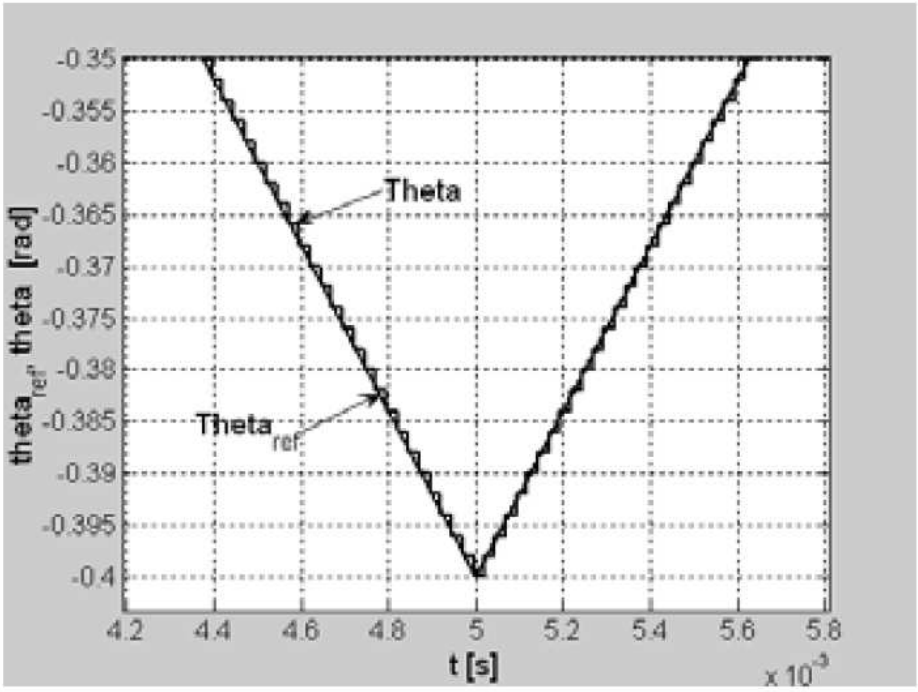


Figure III. 23: Simulation results of the alternative encoder model (Figure 8.b of [23])

4. HARDWARE DESCRIPTION OF THE MODEL

This chapter of our master thesis is focused on finding a hardware model of STC inverted pendulum system that finally will be implemented in a Virtex II Pro FPGA at RAPTOR 2000 motherboard. For doing this the first step consists on replacing Simulink block from the software model by Xilinx System Generator blocks.

4.1. Simulink hardware description: Xilinx System Generator toolbox.

The Xilinx System Generator toolbox is a tool developed by Xilinx that enables designers to generate synthesizable Hardware Description Language (HDL) files using Matlab Simulink environment. These HDL files can be implemented in many Xilinx FPGAs (including Virtex II Pro).

Using this abstract hardware description toolbox, instead of programming in HDL directly, it has many advantages, e.g.: Generating HDL files faster and with fewer errors, because of the abstraction level of this toolbox; and allowing to use the Matlab/Simulink functions in the test bench.

In this section the software model will be transformed into hardware fixed-point model replacing Simulink blocks by Xilinx Generator blocks, which are included in a specific library. As we said in Introduction section, some blocks are equivalents but not all. Therefore, some engineering solutions are necessaries to find out these block equivalences and this section is focus on these cases because they are more interesting than trivial block equivalences.

4.1.1. INTEGRATORS

There is not an integrator block in Xilinx System Generator library. Therefore, a new equivalent block is necessary. Thinking in numerical integration, which is a set of algorithms that calculate the numerical result of a definite integral, the simplest method is rectangle rule, which approximates the integral by the sum of rectangular areas. This is the method chosen because at 1 MHz frequency, which corresponds to 1 μs of sample period or, in this case, time increment, the method is not very important for obtaining good results.

Using this really small increment of 1 μs the numerical integration is basically a ‘brute force’ method. Therefore, selecting trapezoidal or Simpson’s rule, which are better numerical integration methods than rectangle rule, is not a good idea because they are more difficult to implement in hardware and their results will be similar due to the very small increment used in our model.

The equation that defines Rectangle rule is:

$$\int_a^b f(x)dx \approx \sum_{i=1}^{n} f(a+(i-1)\Delta)\Delta \quad \text{(IV.1)}$$

Where (*a*, *b*) is the integration interval that is divided into *n* equal subintervals or increments named Δ, which value is:

$$\Delta = (b - a)/n \quad \text{(IV.2)}$$

The term (*i*-1) gives the approximation based on the top-left corner, this means that height of rectangles is taken at beginning of sample period, as is shown in Figure IV.1.

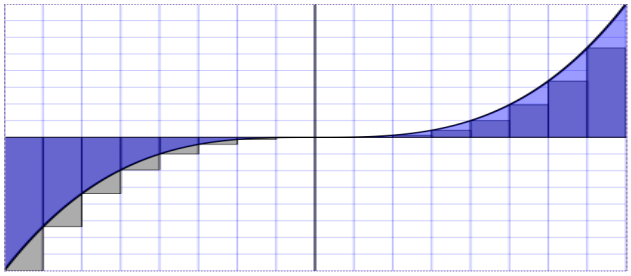


Figure IV. 1: Top-left corner rectangle approximation

After this theoretical explanation, next question is: How do we do a hardware integrator? The key is the summation, which consists on adding, storing, adding, storing... until the index reaches its final value. In hardware language a summation is equivalent to an addition block connected to a register block.

In our model there are four integrators, two for cart position and two for angular position: Acceleration → Velocity & Velocity → Position. Naming ξ to a generic integrator output (Velocity/Position of cart/angular) the Equation IV.1 could be interpreted as:

$$\xi_{CURRENT} = \xi_{LAST} + \Delta\xi \quad \text{(IV.3)}$$

ξ_{LAST} is the last value of ξ that is stored into Register block, but next questions are: What is $\Delta\xi$? How is it obtained? $\Delta\xi$ is the increment or the change in generic variable ξ and for knowing how it is obtained the answers is into differential calculus:

ξ = cart velocity = *v*

In this case the integrator to design is which converts cart acceleration (a_x) into cart velocity (*v*). As is known the acceleration definition is the derivative of velocity with respect to time, from differential calculus point of view, it is the increment of velocity divide by the increment of time. Therefore, the increment of velocity, which is the answer of previous question, is obtained as the multiplication of this input cart acceleration by the increment of time. Finally, Equation IV.4 describes our integrator.

$$a_x = \frac{dv}{dt} \rightarrow a_x = \frac{\Delta v}{\Delta t} \rightarrow \Delta v = a_x \cdot \Delta t \rightarrow \boxed{v_{CURRENT} = v_{LAST} + \Delta v = v_{LAST} + a_x \cdot \Delta t} \quad \text{(IV.4)}$$

ξ = cart position = *x*

This integrator converts cart velocity (*v*) into cart position (*x*). The reasoning applied is the same knowing that velocity definition is the derivative of position with respect to time. See Equation IV.5

$$v = \frac{dx}{dt} \rightarrow v = \frac{\Delta x}{\Delta t} \rightarrow \Delta x = v \cdot \Delta t \rightarrow \boxed{x_{CURRENT} = x_{LAST} + \Delta x = x_{LAST} + v \cdot \Delta t} \text{ (IV.5)}$$

ξ = angular velocity = *w*

This integrator converts angular acceleration (a_θ) into angular velocity (*w*). The reasoning applied is equal to ξ = cart velocity but with angular instead of cart. See Equation IV.6.

$$a_\theta = \frac{dw}{dt} \rightarrow a_\theta = \frac{\Delta w}{\Delta t} \rightarrow \Delta w = a_\theta \cdot \Delta t \rightarrow \boxed{w_{CURRENT} = w_{LAST} + \Delta w = w_{LAST} + a_\theta \cdot \Delta t} \text{ (IV.6)}$$

ξ = angular position = θ

This integrator converts angular velocity (*w*) into angular position (θ). The reasoning applied is equal to ξ = cart position but with angular instead of cart. See Equation IV.7.

$$w = \frac{d\theta}{dt} \rightarrow w = \frac{\Delta \theta}{\Delta t} \rightarrow \Delta \theta = w \cdot \Delta t \rightarrow \boxed{\theta_{CURRENT} = \theta_{LAST} + \Delta \theta = \theta_{LAST} + w \cdot \Delta t} \text{ (IV.7)}$$

Because of sample period in our model is fixed to 1 μs, increment of time (Δt) is equal to this value too. For implementing these integrator equations a gain block is necessary jointly the previously mentioned block (addition and register). In Figure IV.2 is shown an example of an hardware Simulink integrator, in this case cart acceleration (blue input 1 in Figure IV.2) to cart velocity (red output 1 in Figure IV.2) integrator, but remember this model is applied in all integrators, where the unique differences are the input and the output.

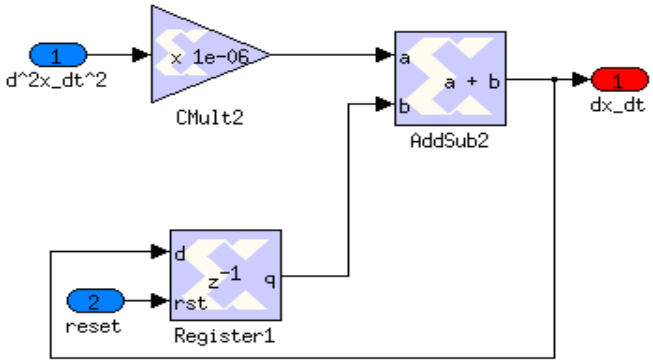


Figure IV. 2: Hardware Simulink schematic for cart acceleration → cart velocity Integrator

4.1.2. ABSOLUTE VALUE BLOCK

In Xilinx System Generator library there is not an absolute value block. In this section a new equivalent block will be found. As our hardware model uses Fixed-point as number representation and its method of representing signed integers is two's complements arithmetic, we known that the most significant bit corresponds to the sign. This bit is the key for doing an absolute value model in hardware. Remember if the sign bit has a value of '0' then the number is positive, on the contrary if this bit is equal to '1' then the number is negative.

The idea for this block consists on: firstly to know if the input number value is positive or negative, then if this value is positive nothing happens (the input goes directly to the output) and if the number is negative it will be multiplied by -1 for converting it in a positive value.

For implementing in hardware this idea is necessary a Slice block, which returns a field of consecutive bits from the input, for obtaining the sign bit. Then connecting it to the select input of a Bus Multiplexer block, just we only need to place a Negate block before port 1 of Mux block for finishing the Absolute Value block. In Figure IV.3 is shown a schematic for this block.

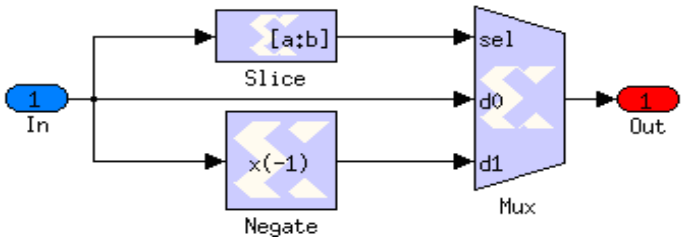


Figure IV. 3: Hardware Simulink schematic for Absolute Value Block

4.1.3. SINE/COSINE BLOCK

In Xilinx System Generator library there is a SineCosine block that enables designers to select its function between sine and cosine but it is not directly equivalent to Sine/Cosine Simulink block. Mathematical functions in hardware context are usually implemented as Look-Up Tables (LUT's), which are data structures with its output values pre-calculated and stored in ROM memory because using LUTs it is cheaper in computation cost (time and memory).

In this case Xilinx System Generator offers a specific LUT for sine or cosine, where it is necessary to fix the input/output width because depending on these values is fixed sine/cosine accuracy. Selecting highest values allowed for Virtex-II Pro FPGA, which are 16 bits for input width and 32 bits for output width (remember our objective is an accurate model and is unimportant the number of Slices or ROM blocks used because we have an entire FPGA for us) the maximum input error is minimized to:

$$\Delta\theta = \frac{2\pi}{2^{16}} \approx 100\mu rad \quad \text{(IV.8)}$$

Equation IV.8 represents the distance (expressed in radians, which is the input unity) between two consecutive array positions in sine/cosine LUT. Such as in a LUT the input value corresponds to an integer (from 0 to 2^{input_width} -1) that indicates the index of the selected array position, where its stored value will be the sine/cosine output, the distance represented by Equation IV.8 is also the maximum input error. This error is enough small to be tolerable in our model.

As has been said, the input of LUT is a positive integer so it is necessary a block that converts the angular position of the pendulum (θ) into an integer. For doing this is used a Gain block that multiplies the angle by $\frac{2^{16}}{2\pi} \approx 1.043 \cdot 10^4 \, rad^{-1}$ and rounds the result to an integer.

Last question to answer is: What happens when the angle is negative? To avoid a negative integer as LUT input it is necessary an Absolute Value block. This is all if you want to model a Cosine block because it is an even function, which means output values are equals for an input and its negative value (mathematically $f(x) = f(-x)$), as is shown in Figure IV.4; but if you want to model a Sine block some extra elements are necessaries (see Figure IV.5) because it is an odd function, which means for negative input values the outputs are the negation of the outputs for positive input values (mathematically $f(-x) = -f(x)$). These extra elements are: a Slice, Mux and Negate block; and they are used to know when the input theta is negative and then negate the LUT Sine output for obtaining a Sine function model in hardware.

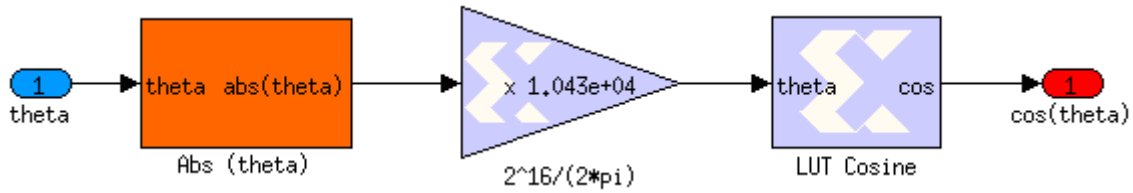


Figure IV. 4: Hardware Simulink schematic for Cosine Block

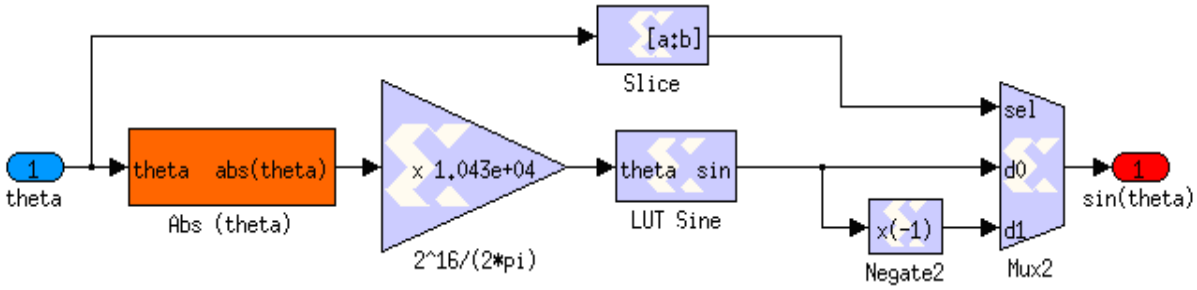


Figure IV. 5: Hardware Simulink schematic for Sine Block

4.1.4. SQUARE (U^2) BLOCK

In Xilinx System Generator library there is not a Square Block but it is easily replaced by a Multiplier block that has both inputs with the same value. This block, which is shown in Figure IV.6, is used in our model to square the angular velocity.

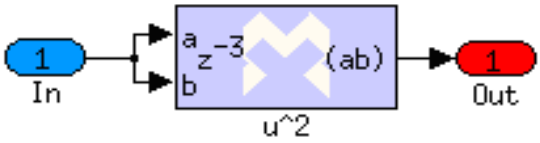


Figure IV. 6: Hardware Simulink schematic for Square Block

4.1.5. MULTIPLIER BY SIGN BLOCK

In our model two times this operation (multiply by sign of variable) is needed: when dynamic friction force is calculated there is a multiplication by the sign of cart velocity, and also when static friction force is obtained it is necessary a multiplication by the sign of input force. There is an optimal hardware implementation for doing this without using a Multiplier block that is shown in Figure IV.7. This way consists on a subsystem block with two inputs: Input 1 is the variable to know its sign (cart velocity or input force depending on case); and Input 2 is the term to be multiplied by the sign of input 1. The behaviour of each element into Multiplier by sign block is:

- Slice block: that is used for knowing the sign of input 1.

- Multiplexer block: that is used to select the final output value depending on the sign bit of input 1. Therefore, its select input signal (named 'sel' in Figure IV.7) is the output of Slice block, its input 0 (named 'd0' in Figure IV.7) corresponds with the input term to be multiplied because if the sign is positive nothing happens; and, obviously, its input 1 (named 'd1' in Figure IV.7) is the negation of input term.

- Negate block: that is used to negate the term providing by input 2 and it is placed just before the Multiplexer input named 'd1' in Figure IV.7.

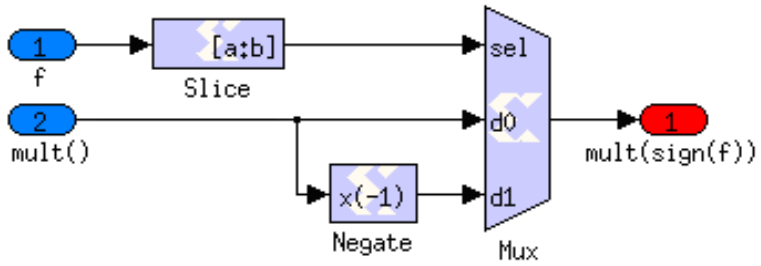


Figure IV. 7: Hardware Simulink schematic for Multiplier by Sign Block

4.1.6. SATURATION BLOCK

Unfortunately, there is not a saturation block in Xilinx System Generator library; thereby, an equivalent hardware schematic was developed. As was said in our software model, the cart velocity is limited by the CKK to ± 0.55 *m/s*. The idea for this design is: firstly, to take the cart velocity input and applying an absolute value to it; after that, comparing this result with the maximum permissible velocity, which is constant a constant block of value 0.55; finally, if cart velocity is out of limits then the output will be plus or minus 0.55 depending on its sign, and otherwise the output will be equal to the input.

As is shown in Figure IV.8, in left side there are a Slice, Negate and Mux blocks that represent an absolute value block, but for optimization (see the output from the Slice block is used in two multiplexer blocks) are not implemented as subsystem.

The Relational block, which is a "is greater than" block, makes the comparison to the maximum permissible velocity defined by a Constant block of value 0.55 (as you can see in Figure IV.8 the constant value is not exactly, because with 32 bits it is impossible to obtain 0.55 using Fixed point notation).

The output of this Relational block is connected to the select input of last multiplexer, which gives the final output (named dx_dt_sat in Figure IV.8) switching between the cart velocity input or ±0.55 depending on its sign bit (see multiplexer named Mux2 in Figure IV.8)

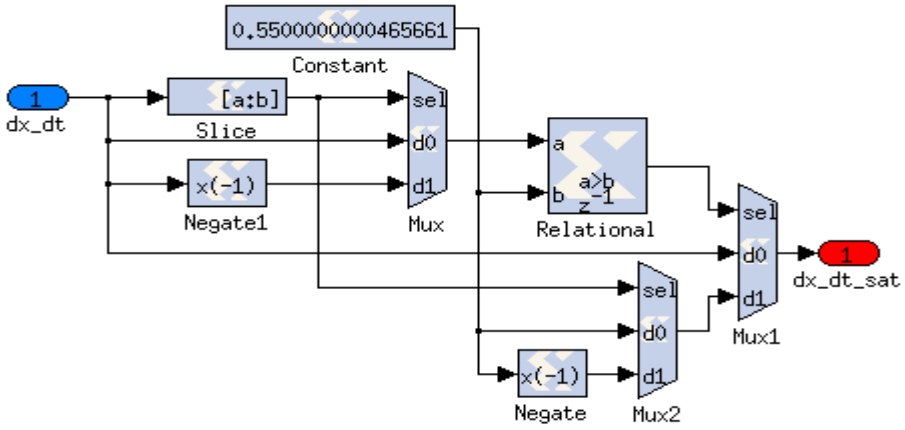


Figure IV. 8: Hardware Simulink schematic for Saturation Block

4.1.7. EXPONENTIAL BLOCK

To obtain the value of dynamic friction force it is necessary an exponential function (remember Equation II.106), such as there is not this type of block in Xilinx System Generator library a Look-Up Table is used in our hardware model. Selecting a ROM block as LUT, which array has 551 positions and its values are pre-calculated using 0.001*m/s* step of absolute value of cart velocity (from 0 to 0.55*m/s*, which is the maximum permissible cart velocity value).

In Figure IV.9 is shown an explanatory screenshot of ROM block configuration, where Depth corresponds to the number of array positions and Initial value vector is the expression that calculates the 551 fixed values with a 0.001*m/s* step. V_s and gamma are constant values defined in the constants script that correspond to Striebeck velocity and the form factor.

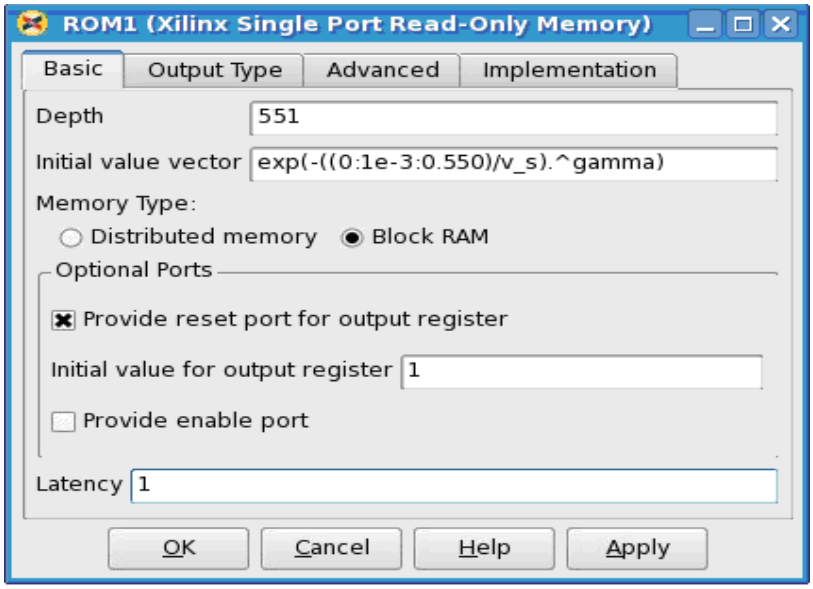


Figure IV. 9: Configuration screen for Xilinx ROM block

In Figure IV.10 is shown the hardware Simulink schematic for this block. Only a gain block is needed to convert the input absolute value of cart velocity into an integer (from 0 to 550) that corresponds to the index of the table and returns the pre-calculated output value.

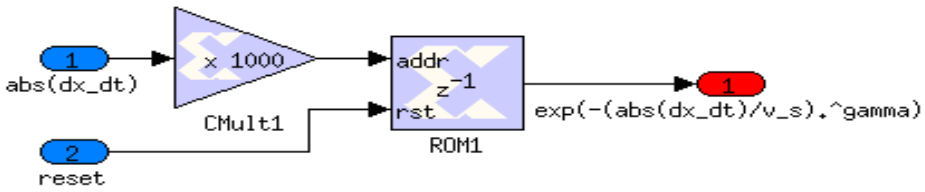


Figure IV. 10: Hardware Simulink schematic for Exponential Block

4.1.8. ENCODER BLOCKS

In this hardware model, like in previous models, every Simulink software block that has a direct equivalence in Xilinx System Generator library has been replaced. For instance: Memory blocks have been replaced by Register blocks; Addition or Subtracting blocks have been replaced by AddSub blocks; Gain blocks by Xilinx System Generator Gain blocks; etc. Therefore, this hardware model is practically the same that software model. The unique difference between them appears in the FSM hardware block, which is called MCode block because it needs an m-file that describes the FSM behaviour instead of the StateFlow Chart used in our software model.

This m-file is basically a translation into Matlab language of the StateFlow Chart. This m-file describes a function that returns two outputs, which are the output bit (‘1’ if the state is Lightness or ‘0’ otherwise) and the next state. This next state output is a difference comparing with the software model and it is necessary because the m-file is basically a “*switch*” function that depends on the last value (one period delayed) of this output, which is named current state and shown in Figure IV.11.

Therefore, the number of inputs is different between software block, which has only angle input (see Figure IV.11). In this Mcode block there are seven inputs: the first one is obviously the angle input; the second one was previously introduced and it is the current state of the FSM; the third one is common to every hardware block that works with Memories/Registers and it is the reset input; the other four inputs, which are threshold, eps, upper limit and lower limit, are related to state transition conditions that are coded using *if else*. For sure, these four values could be defined into the m-file and thereby they will not be Mcode block inputs, but we prefer to implement these variables using Constant blocks, which initial values are defined into the Matlab script of model constants that was explained in the beginning of the third chapter of this master thesis. This way is better because if sometimes it is necessary to change these values it could be done changing only this constant model script.

Remembering the differences between Cart position and Angular position encoder, where are different angle threshold values (see Equation II.49 and II.50), and also the differences between Channel A and B, where the upper and lower limit of the sensor operation area (remember Figure II.14) are different, it is necessary to define eight constants in our Matlab script of model constant that is included as model callback in our Simulink model schematic. These constants are: the upper limit and lower limit for channel A, and the upper and lower limit for channel B into the Cart position encoder; and the same four constant but with different values for Angular position encoder.

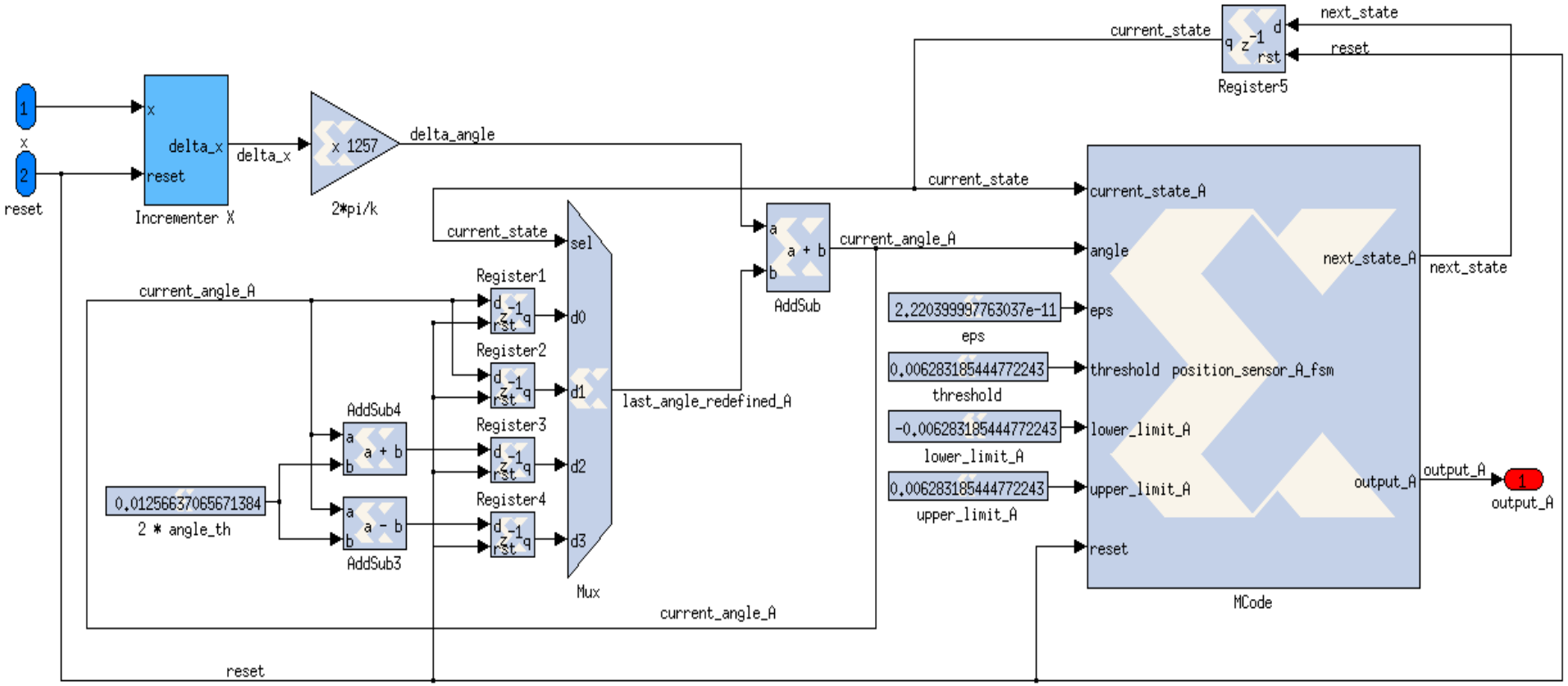


Figure IV. 11: Hardware Simulink schematic for Channel A of Cart Position Encoder

In Figure IV.11 is shown an example of hardware Simulink schematic for encoder model, in this case is shown the Channel A of cart position encoder, but is easily to imagine the rest of them. The differences are:

- *Constant blocks*: that are inputs to the Mcode block. Their values depend on the type of channel and encoder implemented.
- *Gain block*: only appears in schematics of Cart position encoder.
- *Subtract and Constant block*: that must be placed just before the MCode block in Channel B schematic (remember Figure III.10) for modelling the half band distance between sets of LED/photodiode that is necessary to obtain outputs in quadrature.

4.2. Fixed-point notation

As stated at the beginning of this chapter, Xilinx System Generator creates hardware fixed-point models. Therefore, it is necessary a brief explanation about this type of number representation and our criteria for fitting the model with this notation.
Fixed-point notation is a real number representation where the number of digits before and after the decimal/binary (depends on the base used) point is fixed. It is another way to represent numbers in computing instead of floating point, which is more extended but also is more complicated. Fixed-point notation is really simple for representing fractional numbers because, basically, it consists on an integer that is scaled by a factor. Thinking on base 10, because it is simple for humans, the number 34.5678 can be represented using fixed-point notation as the integer 345678 and a scaling factor of 1/10000. Or in other words: the number of digits used by this representation is equal to 6 and the decimal point is on position 4 (34.5678).

To finalize this brief explanation about fixed-point representation, it is necessary to specify that Xilinx System Generator uses base 2 because it is more computational efficient and uses two's complement format for representing negative numbers.

Next step consist on explaining our criteria to fix the number of digits, which are called bits in base 2, for the integer and fractional part of each value in our model. The main rules are:

1. **Every important constant value is defined at least by 32 bits**: This rule is fixed because, as is known, usually depending on the number of bits designated to represent a fractional value this number will be more or less accurate. Of course, after huge effort made to find out the model parameters values, which requires several fractional digits, it is necessary at least 32 bits to obtain an accurate representation of the number value.

2. **To fix the number of bits for integer part apply logarithm base 2 and round up**: If the number is negative, obviously its type is "*signed*" and it is necessary to apply the Rule 4 to avoid problems. But for the moment, forget the sign and apply the logarithm, remember that logarithm function is not defined for negative input values. This result is the number of bits for the integer part in fixed-point notation. Naming *n* to the number of total bits for representing a value and naming *i* to the number of bits for representing its integer part, then we can explain next rule.

3. **To fix the position of binary point subtract *n – i*** : This rule is easily explained, when the number of bits needed to represent the integer part (*i*) is known, the rest of them (*n-i*) are used to represent the fractional part. If this number of bits is not enough for representing pretty accurate a value, then it is necessary to increment *n* (64 bits is enough in this model).

4. **The binary point of a *signed* number must be placed one position less that *unsigned*** : The reason of this rule is simple, when a number is defined as *signed* its MSB is interpreted as sign bit. Therefore, it is necessary one more bit for representing the integer part, as is shown in Equation IV.9; and finally the binary point position in a *signed* number is placed one position less that in a *unsigned* number, as is shown in Equation IV.10.

$$i_{Signed} = i_{Unsigned} + 1 \quad \text{(IV.9)} \quad \rightarrow \quad n - i_{Signed} = n - i_{Unsigned} - 1 \quad \text{(IV.10)}$$

For better understanding of these rules an example is shown. For instance, the model parameter *V_off* is taken. In Figure IV.12 is shown the Xilinx Constant Block for *V_off* : firstly, it is necessary to select the data type, in this case, the offset voltage is always positive so it is marked Unsigned; Secondly, using the constant values defined in our Matlab script of constants that is preloaded (remember Table 4 for *V_off* value and *length_bits_default* is 32 as was said in Rule 1) is filled in. Note that the Matlab function *"ceil"* is used in field called *Binary point* to round up the logarithm base 2 of *V_off* value.

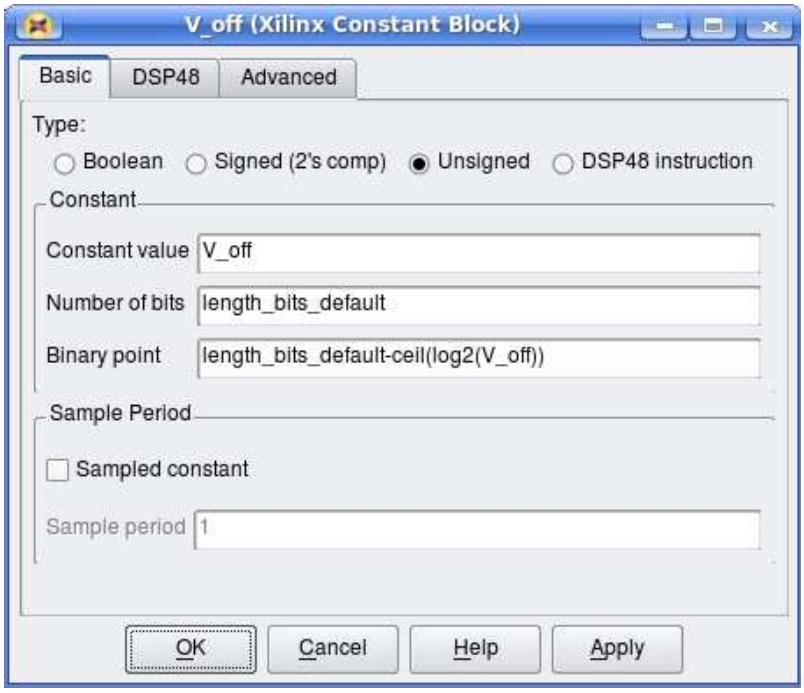


Figure IV. 12: Xilinx Constant Block for parameter *V_off*

At this point, is known how converting constant values into fixed-point notation, but what about variable values? The answer is found thinking on their maximum absolute values and applying the previous rules to these values because if the number of bits is fixed to the maximum value never will be overflow problems.

Now the problem lies in finding these maximum values, some of them are known but others not. The solution for the latter consists in simulating our software model and getting the required values. For instance some variable values of our STC inverted pendulum model are going to be analysed:

- **Cart position (*x*)**: Its maximum absolute value is less than 0.5*m* because the CKK limits the cart displacement. Therefore, such as cart position could be positive or negative, *x* will be FIX 32_31, which means Signed type (FIX), number of bits of 32 and binary point on position 31st.

- **Cart velocity (*dx_dt*)**: It is FIX 32_31 because its maximum absolute value is 0.55*m/s*, which is also limited by CKK.

- **Cart acceleration ($d^2x\_dt^2$)**: It is FIX 32_26 because its maximum absolute value is 18.39 m/s^2 taken from software model simulation. In the Figure IV.13 are shown the results of the cart dynamics obtained by the software model in the Simulink simulation. These results are used for checking that the maximum absolute values for the cart position and velocity (second and third subplot in the Figure IV.13) are correct and also for obtaining the maximum absolute value for the cart acceleration (the last subplot in the Figure IV.13).

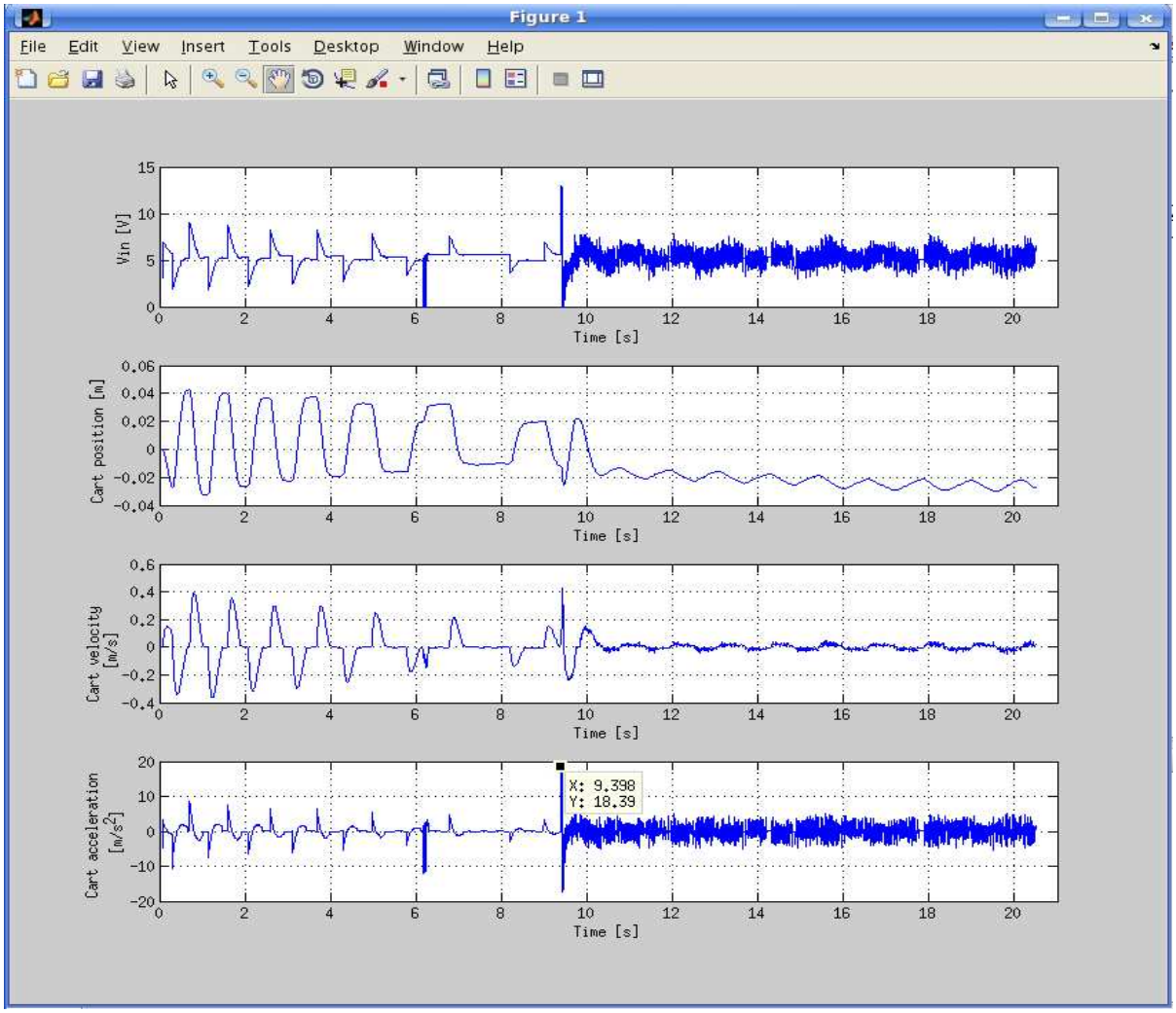


Figure IV. 13: Cart dynamics results obtained by the simulation of the software model

- **Angular position (θ)**: It is FIX 32_28 because its maximum absolute value is 2π, thereby, 3 bits are needed for integer part and one more for sign bit.

- **Angular velocity ($d\theta\_dt$)**: It is FIX 32_26 because its maximum absolute value is around 3/4 loops per second. Therefore, 5 bits are needed for integer part and one more for sign bit.

- **Angular acceleration ($d^2\theta\_dt^2$)**: It is FIX 32_24 because its maximum absolute value is 83.21 rad/s^2 taken from software model simulation. In the Figure IV.14 are shown the results of the pendulum dynamics obtained by our Simulink software model in the simulation. In the same way that the cart dynamics, these simulated results are used for checking that the maximum absolute values for the angular position and velocity (second and third subplot in the Figure IV.14) are correct and also for obtaining the maximum absolute value for the angular acceleration (the last subplot in the Figure IV.14).

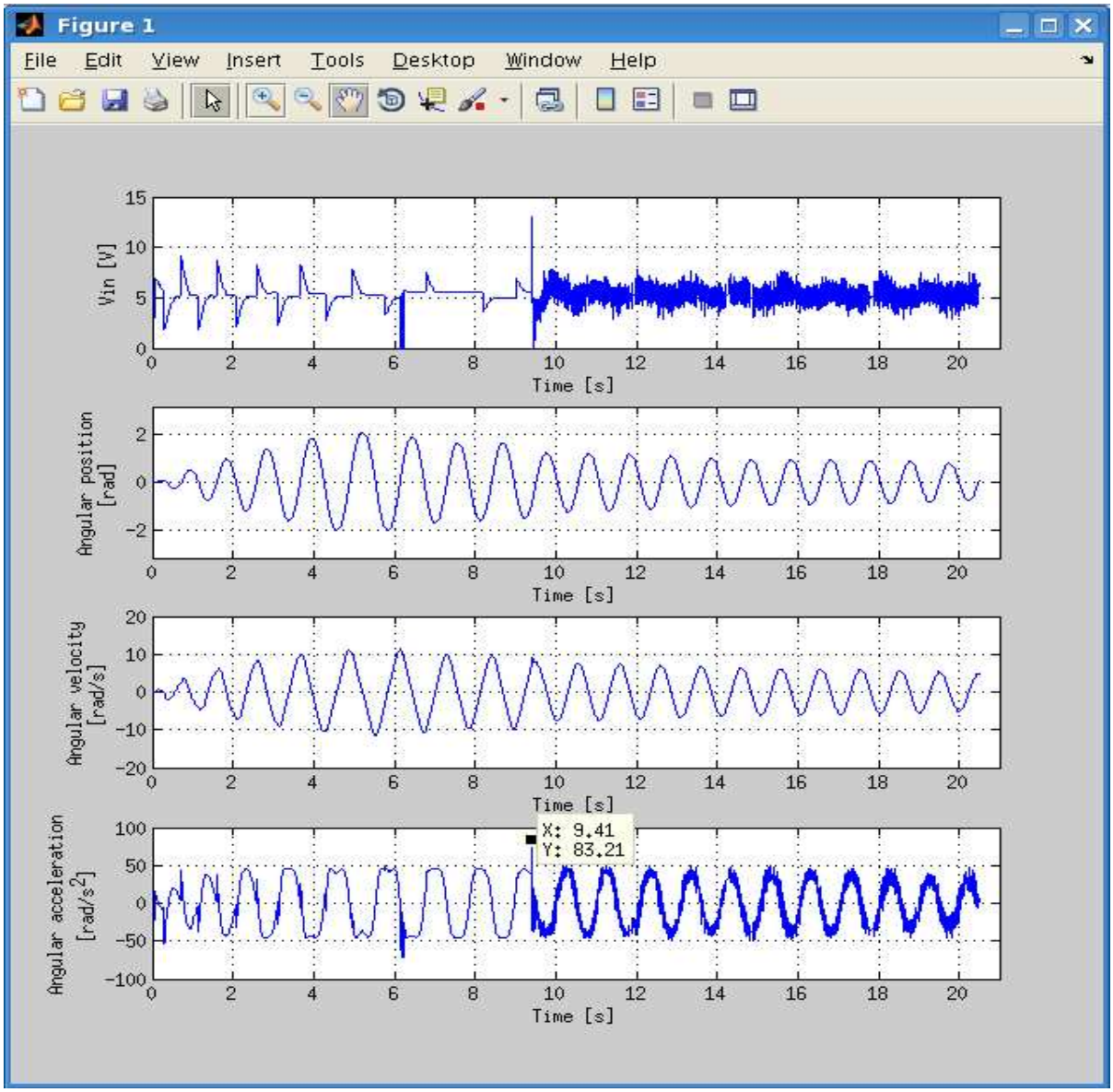


Figure IV. 14: Pendulum dynamics results obtained by the simulation of the software model

Note that every wire in our hardware model needs to be defined in fixed-point notation that represents its variable value. In this documentation it is not shown all because it could be too long, but the idea is to find out the maximum absolute value for each variable. These values are sometimes well known (e.g. the maximum absolute value of the cart position) and sometimes they are determined using Simulink simulations with our software model of the STC pendulum. Next step is to show the process used for implementing this hardware model into a Xilinx Virtex II Pro FPGA.

4.3. Netlist and bitstream creation for FPGA implementation: Hildegart

The objective of this section is to explain how to obtain a bitstream file that implements our hardware model into the Xilinx Virtex II Pro FPGA placed on the RAPTOR 2000 motherboard. As stated previously, System Generator toolbox of Simulink can help us to do it. In Figure IV.15 is shown its design flow

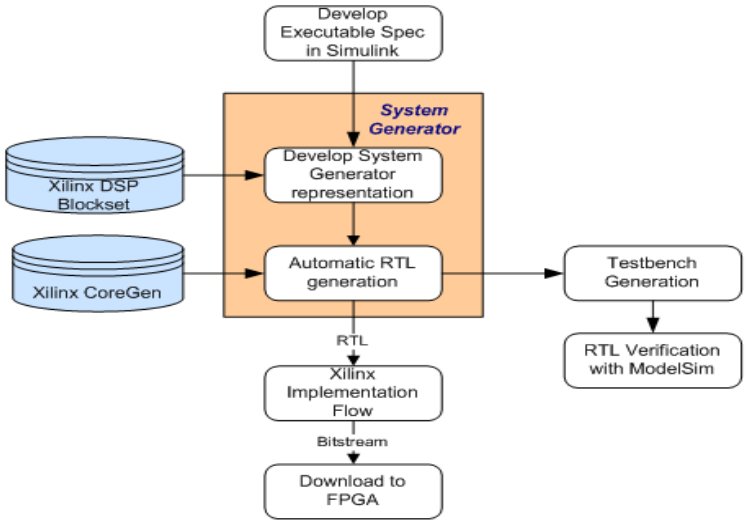


Figure IV. 15: System Generator design flow [24]

At this point we are into the orange box of System Generator in the first box shown in Figure IV.15. Next step is to create a RTL (Register Transfer Language) file, which is hardware description language very close to assembly language, in our case a VHDL file. Although it is possible to create a VHDL file directly using System Generator toolbox, this file will not work into the RAPTOR 2000 motherboard (some extra information is needed). Therefore, this step is divided in two parts:

1. Compile our Simulink hardware model to obtain a NGC netlist, which contains our logical design data and constraints and it is an intermediate step after VHDL conversion.

2. Create a VHDL file starting from this netlist but including extra computing lines that are necessaries to integrate this model into RAPTOR 2000 interface.

To do this first step is necessary to know that every Simulink schematic that uses Xilinx System Generator toolbox needs a block placed on the diagram called System Generator Token, which is shown in Figure IV.16, because it specifies the target netlist, device, performance targets and system period necessaries for the netlist creation.

Figure IV. 16: System Generator Token block

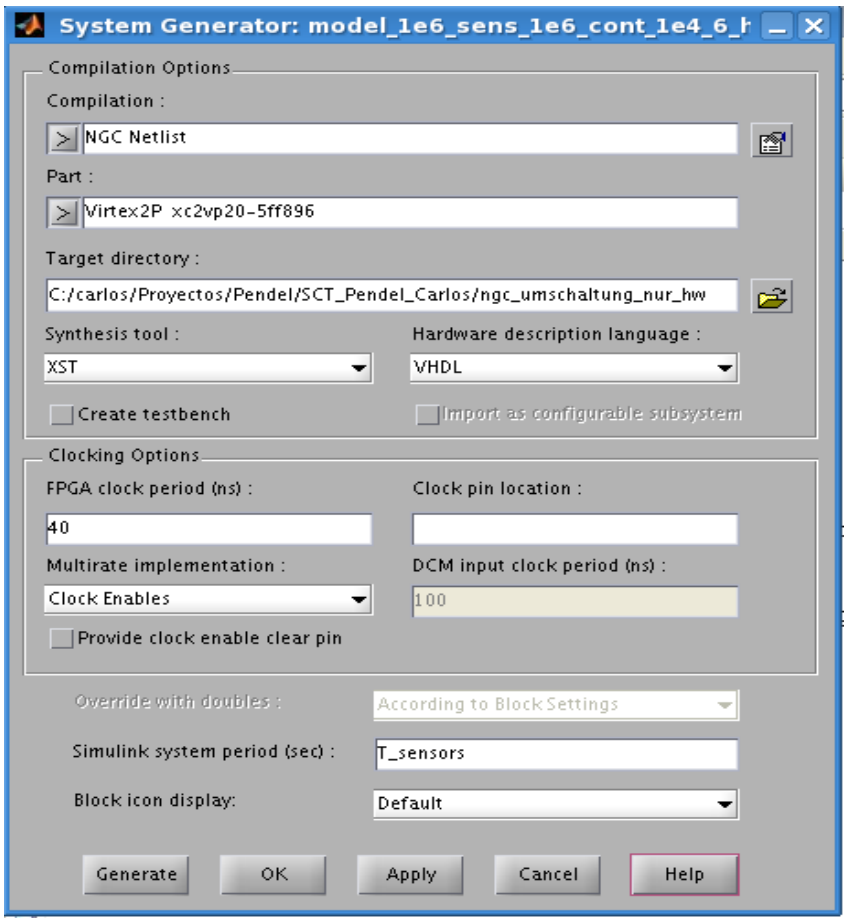


Figure IV. 17: System Generator Token properties

When every field of the properties screen of this block, which is shown in Figure IV.17, is filled out with suitable values. In our case: NGC Netlist into compilation field; Virtex2P xc2vp20-5ff896 as FPGA model; directory root where the netlist will be created; XST as synthesis tool; VHDL is the hardware description language used; the FPGA clock period in nanoseconds; and the Simulink system period, which is set by a constant defined into our Matlab script of constants that is equal to 1 microsecond. Then the button Generate is clicked and the Netlist is created. And the device utilization and timing summary is shown below:

Selected Device:	**2vp20ff896-6**	
Number of Slices:	**6359 out of 9280**	**68%**
Number of Slice Flip Flops:	**1179 out of 18560**	**6%**
Number of 4 input LUTs:	**11588 out of 18560**	**62%**
Number used as logic:	**11399**	
Number used as Shift registers:	**189**	
Number of IOs:	**23**	
Number of bonded IOBs:	**0 out of 556**	**0%**
Number of BRAMs:	**55 out of 88**	**62%**
Number of MULT18X18s:	**24 out of 88**	**27%**

Timing Summary:

Minimum period: 77.525ns (Maximum Frequency: 12.899MHz)
Minimum input arrival time before clock: 0.234ns
Maximum output required time after clock: 0.374ns
Maximum combinational path delay: No path found
Timing constraint: Default period analysis for Clock 'clk'
Clock period: 77.525ns (frequency: 12.899MHz)

Delay: 77.525ns (Levels of Logic = 431)

As seen in the first summary FPGA utilization is roughly 60%. Therefore, if it is necessary a higher accurate hardware model, for instance setting the constant model parameters with 64 bits instead of 32, there is no problems. And as seen in the timing summary the delay is 77.525*ns*, thereby, the maximum frequency that our hardware model could work is 12.899MHz. This means that we do not have timing problem because this value is around thirteen times higher than our maximum model frequency value (c.f. the encoder model frequency at section 2.4. Modelling encoders).

This netlist file, which describes our pendulum model and has extension ".ngc", is included into a VHDL file, which contains the necessary interfaces for communication with the rest of the system (e.g. RAPTOR 2000 or PC), using a software called V-MAGIC[25]. This software is an API written in Java that provides an automatic code generation for VHDL and it has been developed by SCT department. There are three interesting output files after V-MAGIC execution: an m-file, which will be needed for FPGA in the loop simulations when the hardware model is implemented into a Simulink S-function block; a UCF file (User Constraint File), which is an ASCII file specifying constraints on the logical model design; and the VHDL file, which is the objective result of the second step described previously.

Last step consists on obtaining the bitstream file that will be downloaded into FPGA with our hardware STC inverted pendulum model. This step is named in Figure IV.15 as Xilinx Implementation Flow and it is done using Xilinx ISE Project Navigator, which is a software that allows FPGA-specific synthesis. There are multiple tasks that this software can do: from the beginning creating a new project and its schematics, simulating their circuits and implementing them into FPGA. But in our case, we only need to create a new project and adding the existing sources with our VHDL model file and its constraints file. Finally, these file compilation into hardware is done clicking right button on Generate Programming File option and then clicking on Run. The bitstream file (extension ".bit") has been obtained.

To obtain the final configuration of our RAPTOR 2000 motherboard it is necessary to download this bitstream into Xilinx Virtex II Pro FPGA of the pendulum module. This module, which is placed adjoining of Controllers module, needs to be connected through four short wires to the digital I/O interface of Controllers module for sending the encoder output signals. The final RAPTOR 2000 motherboard diagram is shown in Figure IV.18.

This section is finished, next will show the results of our hardware model. There are three different tests to do for analyzing these results: the first one consists on testing our hardware model description with Xilinx blocks in Simulink; the second one consists on testing our hardware model with an offline FPGA-in-the-Loop simulation, where our STC inverted pendulum model is implemented in the FPGA but Controllers block is in Simulink (virtual environment); and finally the third one is to test using on-line FPGA-in-the-loop simulation, where both blocks are implemented in FPGA (as is shown in Figure IV.18), running in real-time, and the augmented reality is used to replace the boring output graphs of x and θ by a virtual image of a pendulum.

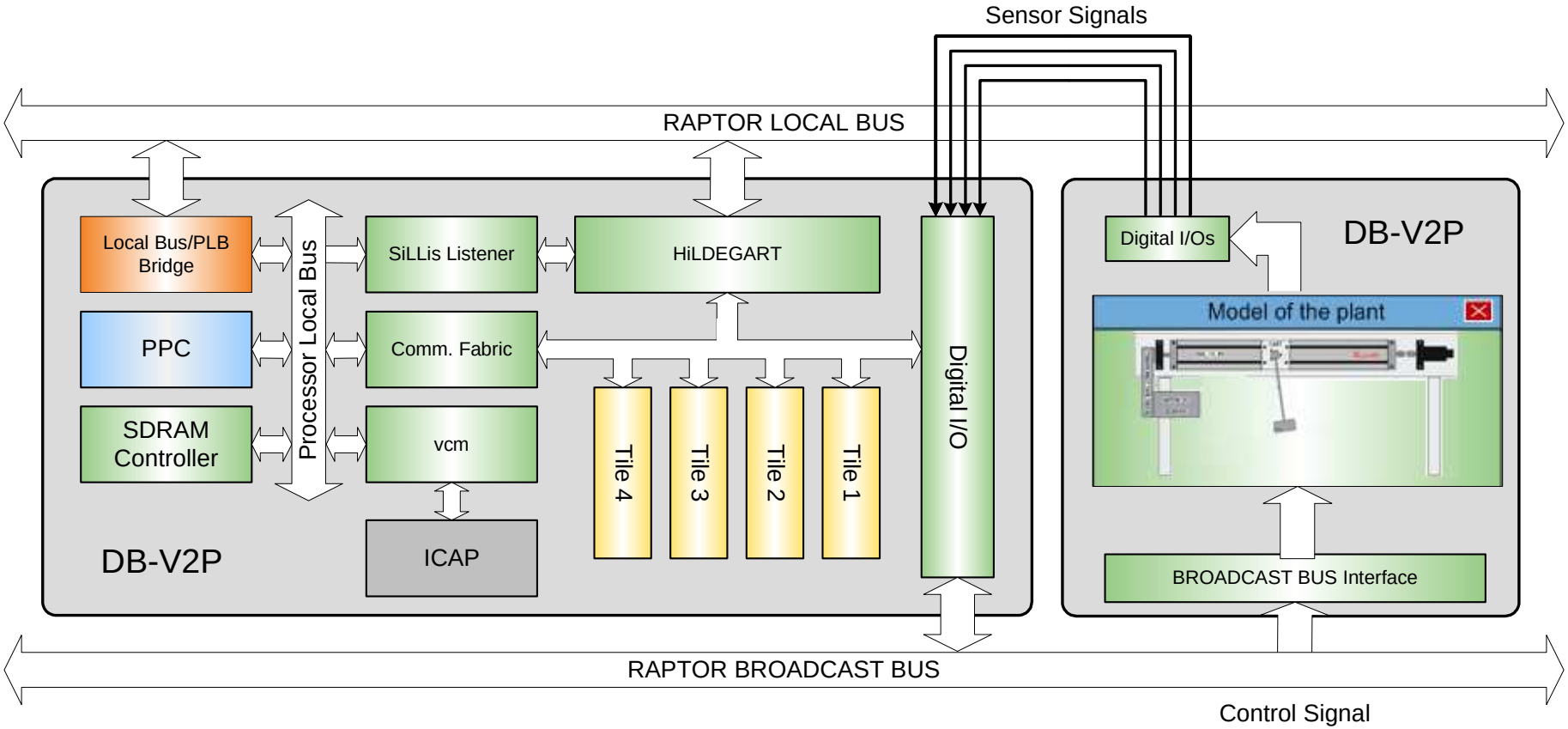


Figure IV. 18: RAPTOR 2000 motherboard diagram of Controllers (left FPGA) and STC inverted pendulum hardware model (right FPGA)

4.4. Testing the hardware model description with Xilinx System Generator

In this section are shown the results obtained by our SCT inverted pendulum hardware model made with Matlab Simulink using the Xilinx System Generator blocks. These results will be compared with our software model and the actual SCT inverted pendulum. As was done in software results section, the Simulink hardware model of encoders will be tested separately because it is better for analyzing their results.

Therefore, in Figure IV.19, are shown the results, which are the cart (upper subplot) and the angular (middle subplot) position, obtained by our hardware model description in Simulink (blue line) when it has as input voltage the signal shown at the lower subplot of Figure IV.19. It is also shown the cart and angular position for the software model (black line) and the actual pendulum (red line) for comparing results.

As seen in Figure IV.19, the simulation lasts for 9.5 seconds, just before the Balance controller takes the control to the end; because a larger simulation only will confuse the reader. Remember that the input voltage signal, which is a training data, from this time to the end is given by the balance controller that achieved the inverted position for the actual pendulum but cannot achieve this for our model because of small differences. Thereby, the response of our both models (hardware and software) is totally different from real, at least for angular position that is not a constant value of $\pm\pi$ (inverted position), see Figure III.11 as instance. Next testing section will show a simulation for whole time because a feedback loop between SCT pendulum model and controller block will exist and the input voltage signal will be suitable to our model outputs.

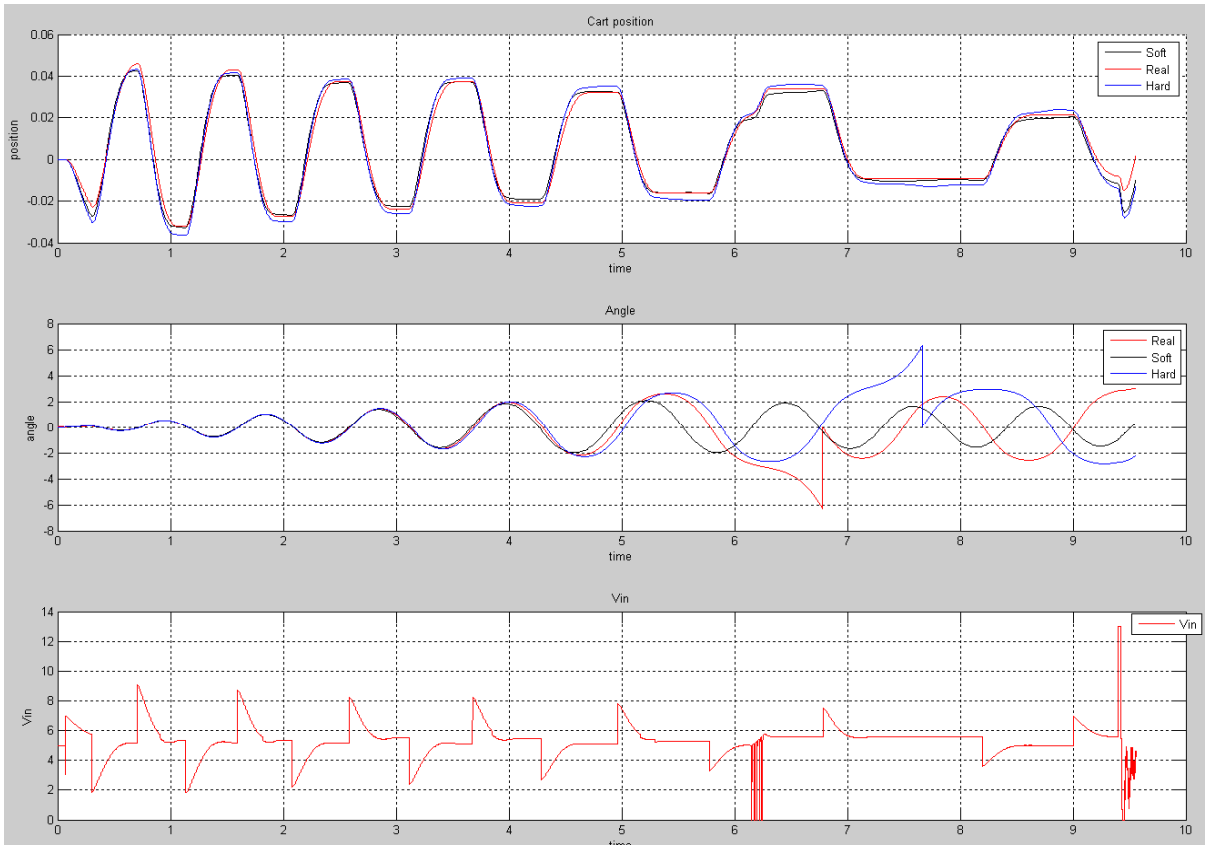


Figure IV. 19: Cart & Angular position for Hard. & Soft. model & real pendulum & input voltage

Analyzing the results we conclude that the hardware model is more similar to the real pendulum than the software model, at least in general terms, because in regard to cart position it is perhaps a bit less adjusted to the real data than the software model (its swings have greater amplitude) but with regard to the angular position can be seen an improvement respect software model, but fails to fit perfectly. If the software model fell short in their amplitudes the hardware model is long. It can be seen as at 8 second of simulation the pendulum is almost in inverted position, one second earlier than the real signal; by contrast the software angular position never achieves the inverted position ($\pm\pi$). Note that the differences between the angular position of the real pendulum and the software model from the sixth second in Figure IV.19 seem to be very large, but in fact they are not, because the real pendulum makes a loop at 6.5 second and the angle of our software model is close to do the loop at this time, and finally it makes the loop in its next chance (at 7.5 second).

This analysis could mislead the reader because is impossible that the hardware realization of the model gives better results than the software implementation because of the quantization effect, which is the difference between the actual value and quantized digital value due either to rounding or truncation (i.e. the loss of accuracy in a value due to the fixed-point notation). Furthermore, there are other differences between our software and hardware model of the SCT inverted pendulum, e.g.: the integrator blocks, the sine/cosine blocks and the exponential blocks. Thereby, our hardware model is quite different to our software model, but these differences, by chance, get our hardware model closer to real pendulum. It could be the other way around and these differences could have moved our hardware model away from the real pendulum, but fortunately it did not happen.

Next step is to analyze the results given by our hardware model for encoders. In Figure IV.20 these results are shown after a simulation of four seconds because this duration is enough for seeing the response of this hardware block.

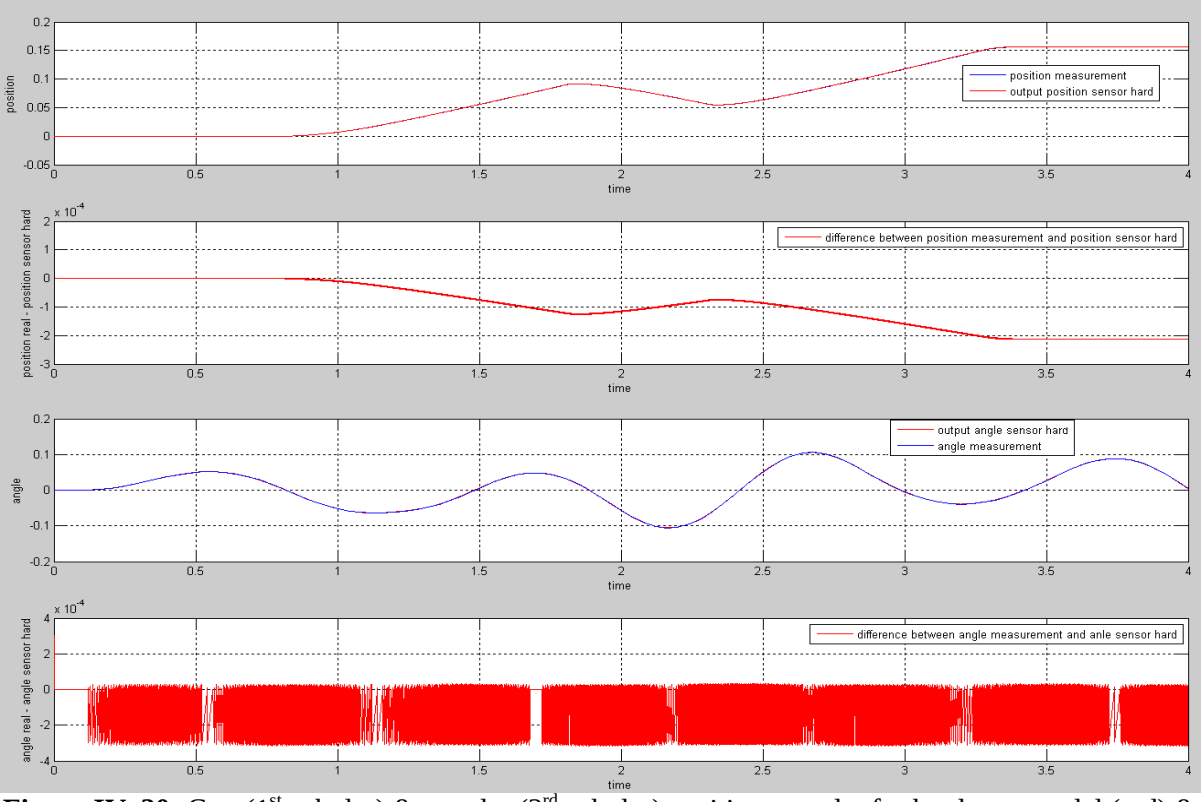


Figure IV. 20: Cart (1st subplot) & angular (3rd subplot) position encoder for hardware model (red) & real data (blue) & their differences (2nd subplot for cart position and 4th subplot for angular position)

Four subplots are shown in this Figure IV.20, the first graph or subplot is the comparison between the output of the cart position encoder and the real data measured; the second subplot shows the difference between them for showing the error produced in our hardware model for the cart position encoder; the third and fourth subplot shows the comparison and difference, respectively, between the output of the angular position encoder and the real data measured.

The hardware model for encoders block is entirely satisfactory because, as is shown in Figure IV.20, both encoders have a response almost equal to the ideal, as can be seen in the graphs that show the difference between the outputs of the sensors and the actual measurements.

It is true that the second subplot shows the same inversely proportional behavior that the cart position encoder shows in the software model. It is natural that it should be so, because it proves that the software and hardware model should give very similar results as long as the bocks of Simulink are replaced by the System Generator blocks that are equivalent. This happens in the model of the encoders but not so in the global model where, as stated above, there are certain blocks that are not exactly equivalent and they introduce some small errors. For example: the integrator blocks, which use a different algorithm for integration that the integrator blocks of Simulink library (c.f. section 4.1.1. Integrators); or the sine/cosine and exponential blocks, which use LUTs that introduce small errors compared to the Matlab/Simulink math functions (c.f. sections 4.1.3. Sine/Cosine block and 4.1.7. Exponential block).

However this small error in the position of the cart due to a very small delay between the data measured and the real data is not a concern because looking the order of magnitude of this error it is around $10^{-4}m$, thereby, it is not problematic and could be ignored.

Before concluding this analysis of the hardware encoder model, note that the size of the difference signal between the real and the hardware encoder model for the cart position is in the order of magnitude of 10^{-4} m and it seems to be an error, because it is one order of magnitude higher than the same result for the software encoder model (remember Figure II.20), which is 10^{-5} m of order of magnitude, but it is not.

The explanation is that the cart position signal taken as data to test this hardware encoder model is one order of magnitude higher than the cart position data taken to test the software encoder model (see Figure II.20 where the cart position signal is between -0.04m and 0.05m). Another proof to test that this hardware encoder model works perfectly is found comparing the difference signal from the software model for the angular position encoder (see Figure II.19) and the difference signal from the hardware encoder model in Figure IV.20, where both maximum values of these signals are less than 0.4mm in absolute value.

4.5. Testing the hardware model using offline FPGA-in-the-Loop simulation or HiLDE

In this section our hardware model is tested using Hardware-in-the-Loop Development Environment (HiLDE), which means that the Design Under Test (DUT), which is our hardware model of the STC inverted pendulum, is implemented in a real FPGA but the test environment is virtual (Matlab Simulink in our case). For doing this test it is necessary an interface that establishes the connection between the DUT and the controllers implemented in Simulink, which is the simulator that computes the test environment and where all data transfers (including the clock signal) are controlled by this simulator allowing the functional verification of the DUT.

Using this FPGA-in-the-Loop simulation, which is a HiLDE simulation where the hardware model is implemented, in this case in a Xilinx Virtex II pro FPGA included in the RAPTOR motherboard, it is tested the functionality of the DUT but it is not tested its timing behaviour because this virtual environment does not work in real time, because of it, this simulation is called *offline*. The final step is going to be an online FPGA-in-the-Loop simulation for verifying the timing behaviour of the DUT, but at the moment the objective of this test is to verify the functionality of our STC inverted pendulum hardware model implemented in a FPGA working with the controllers in Simulink.

The first test consists on verifying that our hardware model of the pendulum system works correctly with the Balance controller. The results are shown in Figure IV.21, where is seen how the pendulum is balanced in inverted position (π), look at the graph above of Figure IV.21 where is plotted the angular position, and how the cart is forced to move four centimetres to left direction (red line in middle subplot of Figure IV.21) at the first second of the simulation. The response to this impulse of our DUT is shown by the black line (cart position) in middle subplot and the blue line (angular position) in above subplot of Figure IV.21. We can see how the Balance controller achieves the equilibrium for this new position of the cart (four centimetres to left of initial position) through its output voltage signal, which is the input of our hardware pendulum model, shown in subplot below of Figure IV.21.

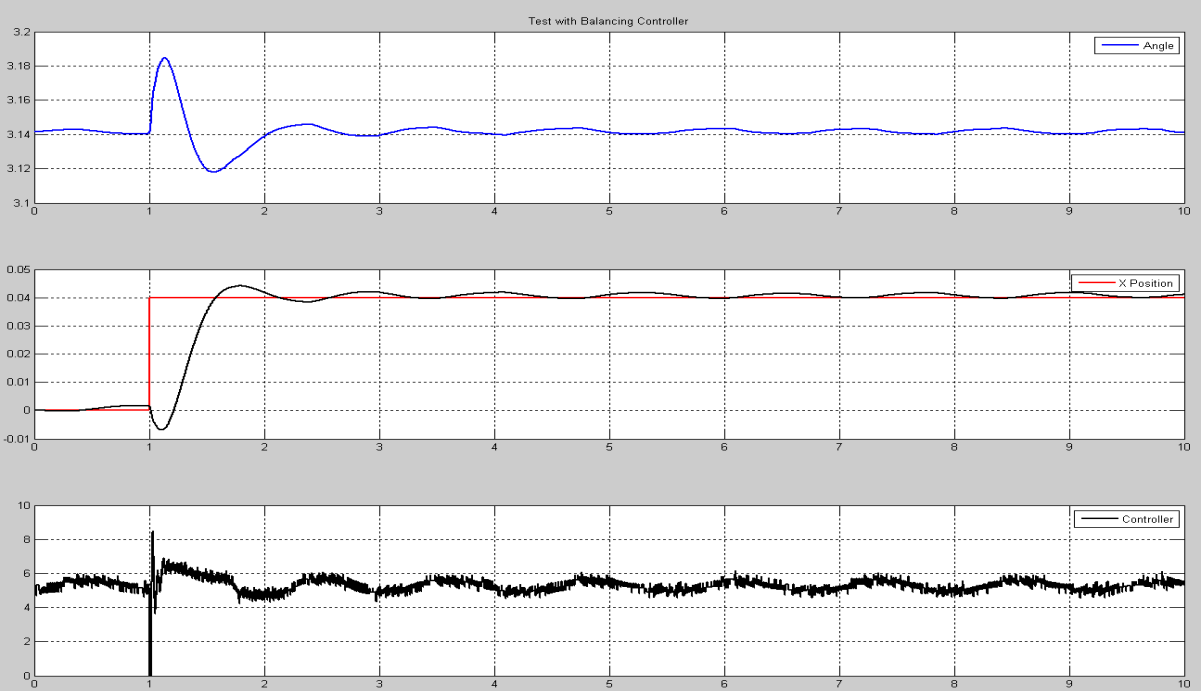


Figure IV. 21: HiLDE outputs for our hardware model of STC inverted pendulum with Balance controller

Analyzing these results we can assert that our model response is correct. The cart position achieves the target position, after one second, and the angular position too. It is true that an oscillation appears in both outputs but this behaviour is normal using the real pendulum; thereby, if it is an error is produced by the controller and not by our hardware model. Regarding the timing behaviour, which needs more than one second to achieve the equilibrium, it is not important in this test because it is not real time: remember that in offline FPGA-in-the-Loop simulation the time is controlled by the simulator for adapting the DUT implemented into the FPGA, which always works faster than Matlab, to the Simulink environment where is placed the Balance controller.

The second test consists on verifying that our hardware pendulum model responds fine with the Swing Up controller. This means that the output voltage signal of this controller, which is the input of our hardware model, gets growing amplitude for the pendulum swings through cart movements from right to left.

The results are shown in Figure IV.22 where we can see how the output voltage signal of the Swing Up controller (black line at the graph below in Figure IV.22) forces the cart to move from right to left (see middle subplot) and this movement causes an increasing amplitude of the angular position that achieves the objective (±π) even more as is shown in subplot above of Figure IV.22. Remember that when both controllers are working together a supervisor block changes between them when some conditions are satisfied, as instance: the angular position is close to ±π or the angular velocity is low enough. But remember the behaviour and design of controllers is not our objective.

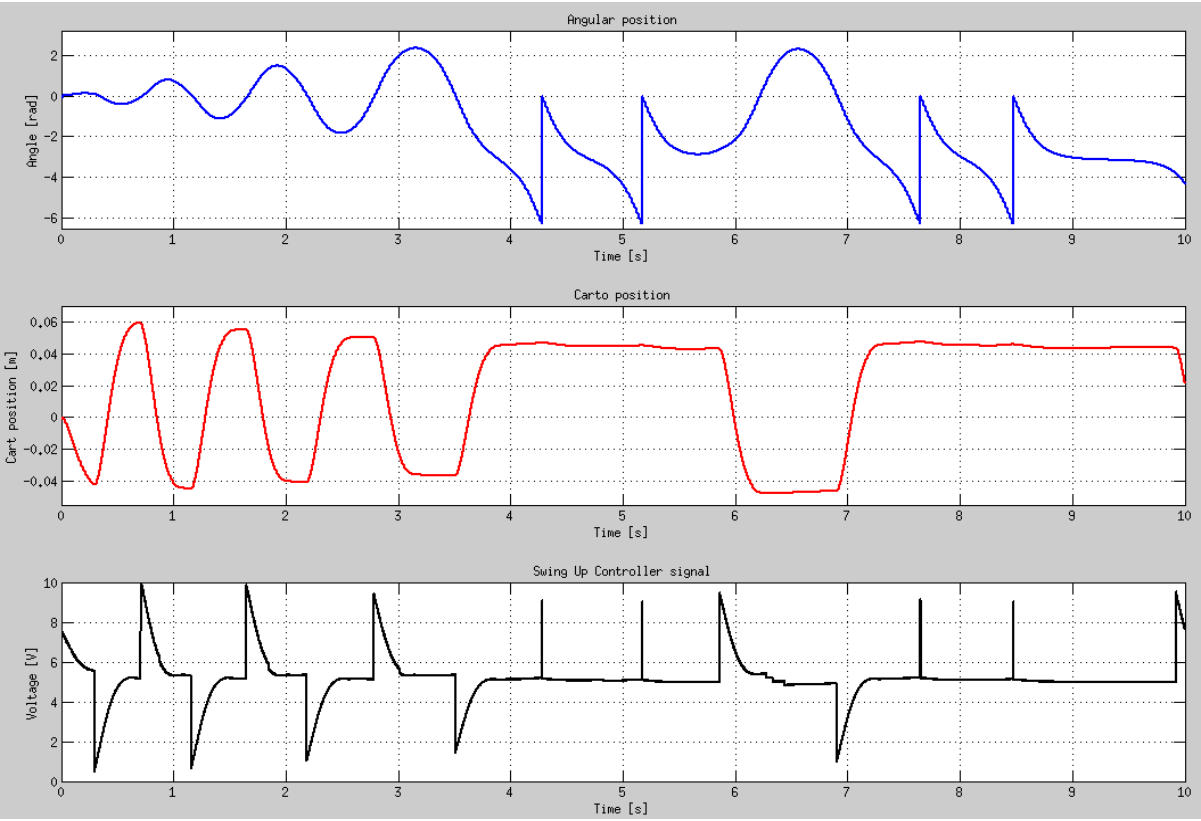


Figure IV. 22: HiLDE outputs for our hardware model of STC inverted pendulum with Swing Up controller

As is shown in subplot above of Figure IV.22 the angular position of pendulum increases its value reaching the inverted position just before the fourth second of this simulation. From this time to the end of simulation the angle of the pendulum still increasing its amplitude, even doing four loops at 4.2, 5.1, 7.6 and 8.5 seconds. Thereby, we can assert that the response of our hardware model of SCT pendulum is correct and this test has been a success.
The last test using HiLDE simulation will be the most important test done so far, it consists on testing our STC inverted pendulum model with both controllers working together. If this test obtains good results it will mean that our model is a good substitute for real pendulum. The results are shown in Figure IV.23, which shows four different graphs that are listed from top to bottom: angular position, cart position, Swing Up controller voltage signal and Balance controller voltage signal.

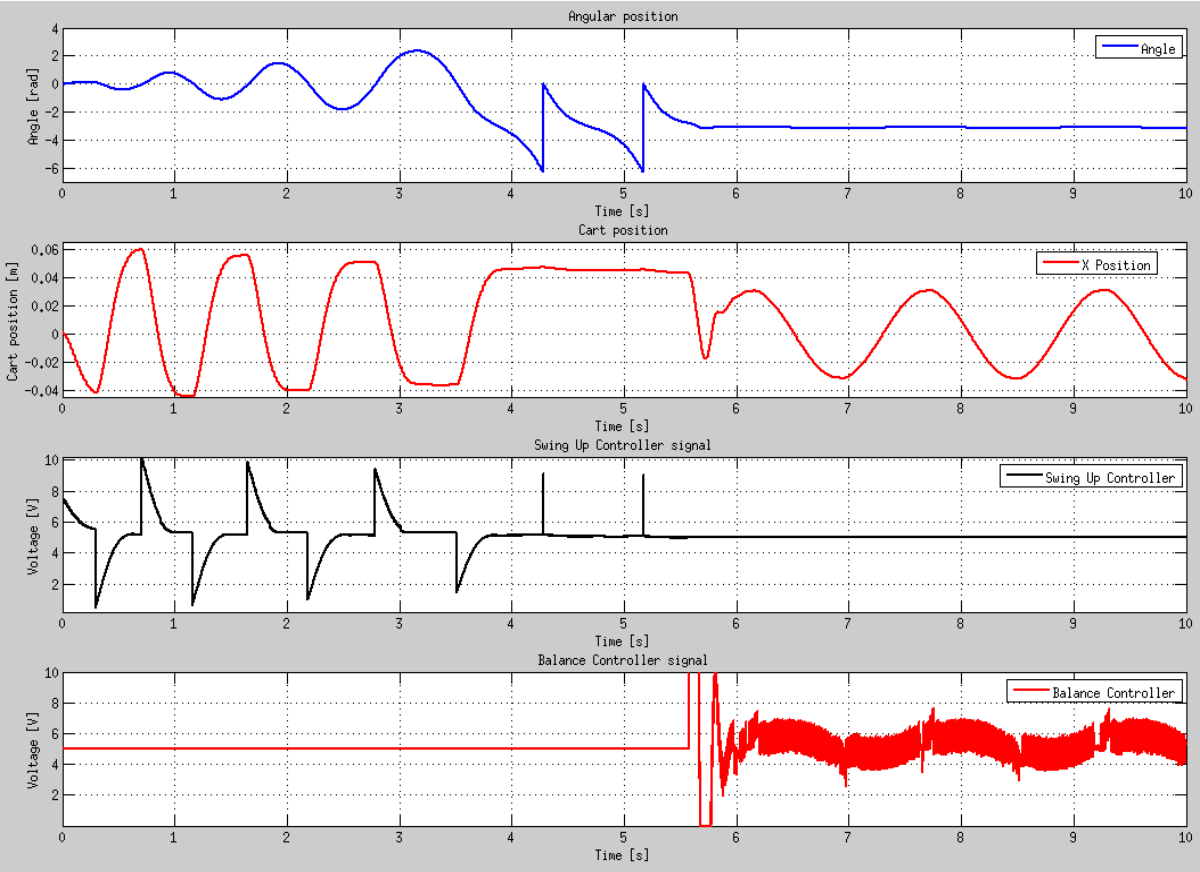


Figure IV. 23: HiLDE outputs for our hardware model of STC inverted pendulum with both controllers

The first positive thing to note is that our pendulum model is balanced in inverted position by the controllers as we expected. This is seen in the angular position subplot of Figure IV.23, where from 5.5 seconds to the end its value is -π.

Unfortunately, the behaviour of our pendulum model using both controllers is different as we expected, particularly, when the Balance controller takes the lead (from 5.5 seconds to the end of the simulation). As is shown in the cart position subplot of Figure IV.23, the oscillations of the cart from 5.5 seconds of the simulation are too large, more than ten times compares to the real pendulum oscillations (see Figure IV.24 from eighth second).

This behaviour is due to the Balance Controller signal, which is different to the signal shown in Figure IV.21. The explanation is that the first HiLDE test (Balance Controller test alone) and the last HiLDE test (with both controllers) are testing different situations, the first one consists on testing how the Balance Controller achieves the equilibrium when the pendulum is in the inverted position and, suddenly, the cart is moved 4 cm to the left; meanwhile, the other one consists on testing how the Balance Controller achieves the equilibrium starting from a swing up situation, which is not the inverted position.
On the other hand, the signal provided by the Swing Up Controller in the test with both controllers, which results are shown in Figure IV.23, corresponds with the signal supplied by this controller when we tested the Swing Up Controller alone (c.f. Figure IV.22 until 5.5 seconds). This behaviour is the same that we expected, because in this case, unlike the Balance case, both tests start from the same situation (the pendulum rest position) and make the same goal that is to swing up the pendulum.

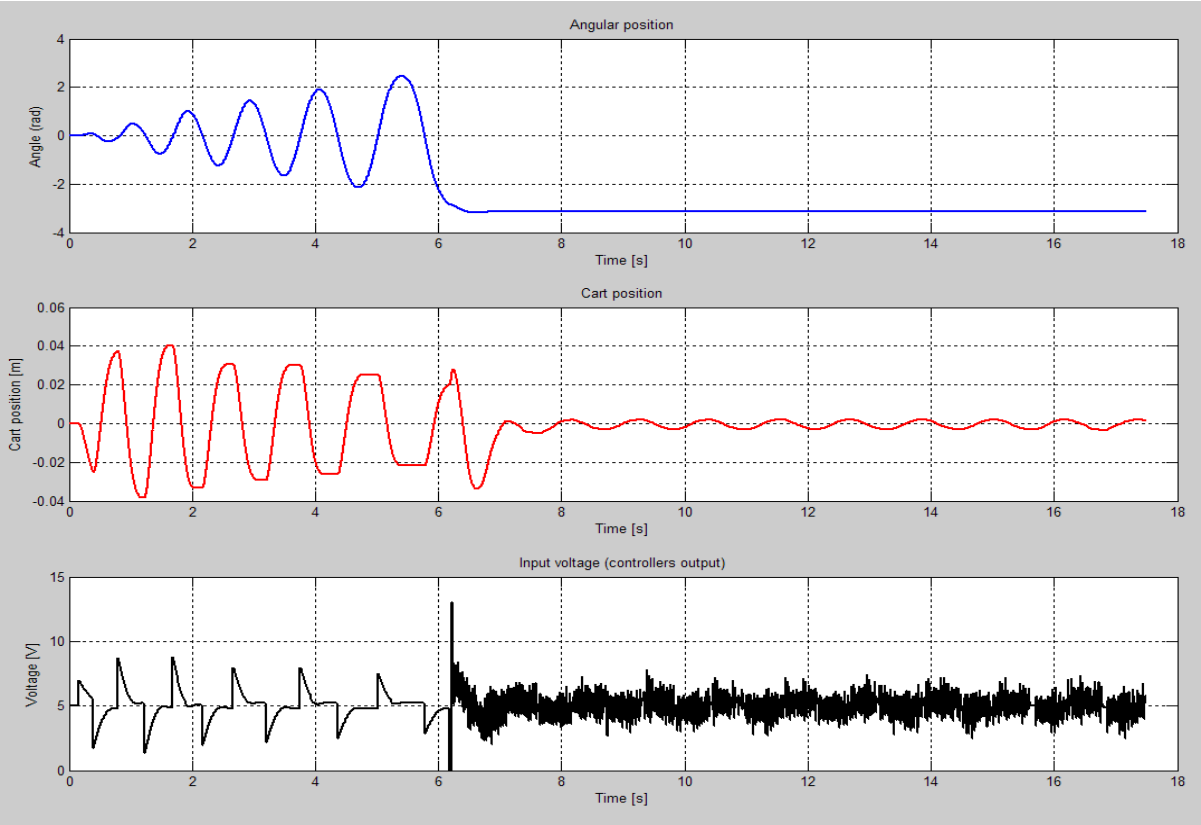


Figure IV. 24: Angular (top subplot) and cart (middle subplot) position signals of the real pendulum and controllers signal (bottom subplot) taken from the real system

The last comparison in this analysis is between the results obtained by the real SCT inverted pendulum (Figure IV.24) using these controllers and the results obtained by our hardware pendulum model (Figure IV.23).

The first point to note is the different input signal that the controllers supply to the pendulum in both cases. This signal should be the same, but as is shown in Figure IV.24, for the real pendulum it is an 80% of the input signal supplied by the Swing Up Controller to our hardware model. The cart movement has smaller oscillation than in Figure IV.23 in consequence of this smaller signal, and thereby, the amplitudes of the pendulum swings are smaller too. This is the reason why the angular position of our hardware model achieves the $-\pi$ value at fourth second (c.f. Figure IV.23) and the real pendulum has to wait until the sixth second (c.f. Figure IV.24).

The second point to emphasize is the moment of the switch between controllers, i.e. when the pendulum is balanced in inverted position. In the real pendulum this moment occurs at the sixth second of test whereas the hardware model achieves the balanced position at 5.5 seconds. This is explained also by the different voltage signal supplied by the Swing Up Controller in each case. As a curiosity to emphasize that the hardware model makes two loops (at 4.2 and 5.1 seconds in Figure IV.23) because of the excessive force that is supplied to the cart, and the reason why we know that it is an excessive force is because the pendulum does not meet the controllers conditions for doing the switch between them. It can be checked looking the Balance Controller signal in the Figure IV.23 at this period of time (from 4 to 5.5 seconds).

The last point to note is the differences between the behaviour of the real SCT inverted pendulum and our hardware pendulum model when the Balance Controller takes the lead. As we stated previously, the cart movements are ten times larger in our model response (Figure IV.23) than the same oscillations of the cart in the real pendulum, which are shown in Figure IV.24. The first idea that comes to mind is to hold the different input voltage signal responsible for this behaviour, but comparing the voltage levels from both input signals we can give the same explanation of the Swing Up Controller signal, where the signal for the real pendulum is an 80% of the input signal supplied by the Swing Up Controller to our hardware model.

However, there are some differences between these signals: the input signal from our hardware model (Figure IV.23) is more compact than the other, which is more noisy (i.e. there are more peaks above and below the mean of the signal); and the input signal from our hardware model spends a quarter of second to stabilize the pendulum (from 5.5 to 5.75 seconds in Figure IV.23) supplying the maximum voltage (10V) at the beginning and after the minimum voltage (0V) to the cart, while this stabilization period in the real SCT inverted pendulum is really fast (at 6.1 seconds in Figure IV.24).

The explanations for these differences between the real SCT inverted pendulum and our hardware pendulum model can be summarized in two aspects: the first one is well known at this point of the master thesis, our hardware model is a bit different from the real pendulum; and the second one needs to be demonstrated with another test, the timing behaviour of these hardware model tests (HiLDE) is not real time and the results obtained by the real pendulum, which are shown in Figure IV.24, are in real time and, perhaps, this can be the cause of the quarter of second spent by our hardware model to stabilize the pendulum in the HiLDE test.

Digging deeper into the causes of the differences between our hardware model and the real pendulum, we want to analyze it from the beginning: the first point that can be problematic is the linear simplification applied to the mechanical part of our system when we modelled it (c.f. section 2.2. Modelling Power Driver Controller, Motor AC, Timing Belt and Spindle), because perhaps the error committed by this linearization can not be ignored; next problematic point is that some parameter values of our non linear model, which was successfully obtained in chapter 2. Modelling the pendulum could not have been adjusted 100% to the real values and, obviously, this generates differences between the non linear model and the real pendulum; and the final problematic point is that does not exist a directly equivalence of every Simulink block in the Xilinx System Generator library, and, therefore, some small errors are generated when the software model is converted into a hardware model (e.g. integrator blocks, sine/cosine blocks and exponential blocks).

On the other hand, after finishing the HiLDE tests we can compare them with the Xilinx System Generator test. The first point to emphasize is the differences between both input voltage signals, which is supplied by the controllers, because the input voltage signal of the Xilinx System Generator test, which is shown in Figure IV. 19 and it is supplied by the Swing Up controller, is around a 30% lower than the same input signal supplied by the Swing Up controller in the HiLDE test, which is shown in Figure IV. 22. This can explain the different behaviour of the cart and the angular position in both tests, as is normal the cart and angular position of HiLDE test have higher values (i.e. higher cart oscillations and thereby higher angle oscillations) than in System Generator test.

Now the question is why these input signals are different if the controllers and the hardware model are the same in both tests, the answer is easy, the input voltage signal of the System Generator test is the training data taken from an experiment with the real pendulum, meanwhile, the input voltage signal of the HiLDE test is supplied online to our hardware

pendulum model by the controllers. Therefore, the controllers generate different input signals because of the small differences between the hardware model and the real pendulum, and, obviously, we can not compare this test directly.

Storing the output voltage signal, which is generated by the controllers in the final HiLDE test, in the Matlab Workspace and using it as input voltage signal for our pendulum model in a new Xilinx System Generator test, we can conclude that there is no difference between our model behaviour at the HiLDE and Xilinx System Generator tests. The results of the new Xilinx System Generator test are exactly the same that the results of the HiLDE test, which are shown in Figure IV.23, and it means that our DUT (i.e. our hardware model of the SCT inverted pendulum) has passed the functional verification test.

Next section consists on testing our hardware model using online FPGA-in-the-Loop simulation in real time because it is the final verification for our DUT. Furthermore, we want to test if the quarter of second that our hardware model spends to stabilize the pendulum in the HiLDE test is because of it is not real time. And we will use this final test to verify the compatibility between the Augmented Reality technology and our hardware model implemented in the RAPTOR motherboard in real time.

4.6. Testing the hardware model using online FPGA-in-the-Loop simulation or HiLDEGART and through Augmented Reality

In this section our hardware model is tested using an online environment called HILDEGART, which is a Hardware-in-the-Loop Development Environment for Generic Active Realtime Testing. This means that the test environment is in the hardware architecture (not in software as HiLDE), therefore, the DUT or our hardware model of the STC inverted pendulum, which is implemented in a real FPGA within a RAPTOR module, is tested in real time with the controllers implemented in other real FPGA within a different RAPTOR module as is shown in Figure IV.18. This type of test allows us to check the real time behaviour of our DUT using the Graphical User Interface (GUI) contained at HiLDEGART. And this test is also used to check the compatibility between our hardware model and the Augmented Reality in real time.

As we stated in introduction section, the term Augmented Reality is a technology that provides a direct or indirect view where there are real and virtual elements mixed. In our case, a virtual image of a pendulum is going to appear at the screen inside the real image taken by the webcam.

It is an interesting concept for us because the hardware pendulum model is implemented in a chip; therefore, the “magic” view of an inverted and balanced pendulum will replace the boring output graphs (cart and angular position). The idea is to show at the screen a virtual image of a pendulum, which will move in real time as the same way that the hardware model will indicate.

The operation of the Augmented Reality consists on the pattern recognition in the image taken by the webcam and including a virtual three-dimensional model (marked as B in Figure IV.25 and IV.26) associated with that pattern inside the real image. In our case, this pattern is a particular symbol printed on a card that is marked as A in Figure IV.25 and IV.26.

This is a real time application that the SCT department has developed before we started this master thesis; therefore, we only have taken his work and we have tested it in conjunction with our hardware model of the SCT pendulum to verify that it was compatible.

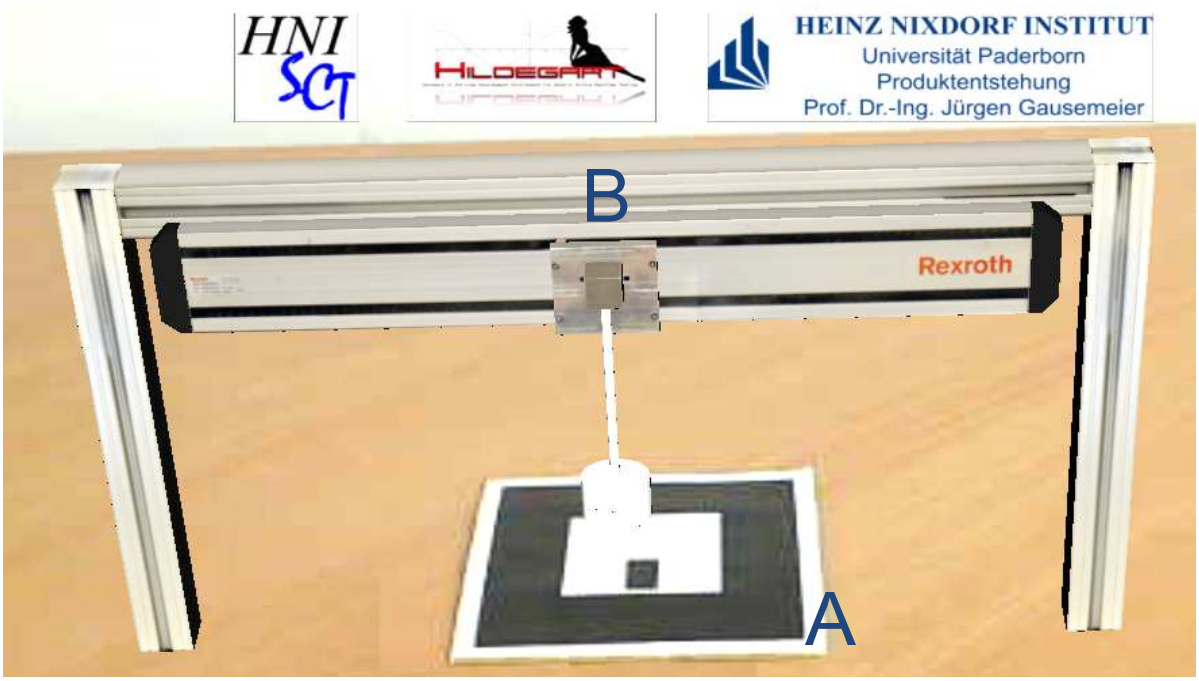


Figure IV. 25: Picture of the PC screen that shows the virtual image of the pendulum (B) at rest and its associated pattern (A)

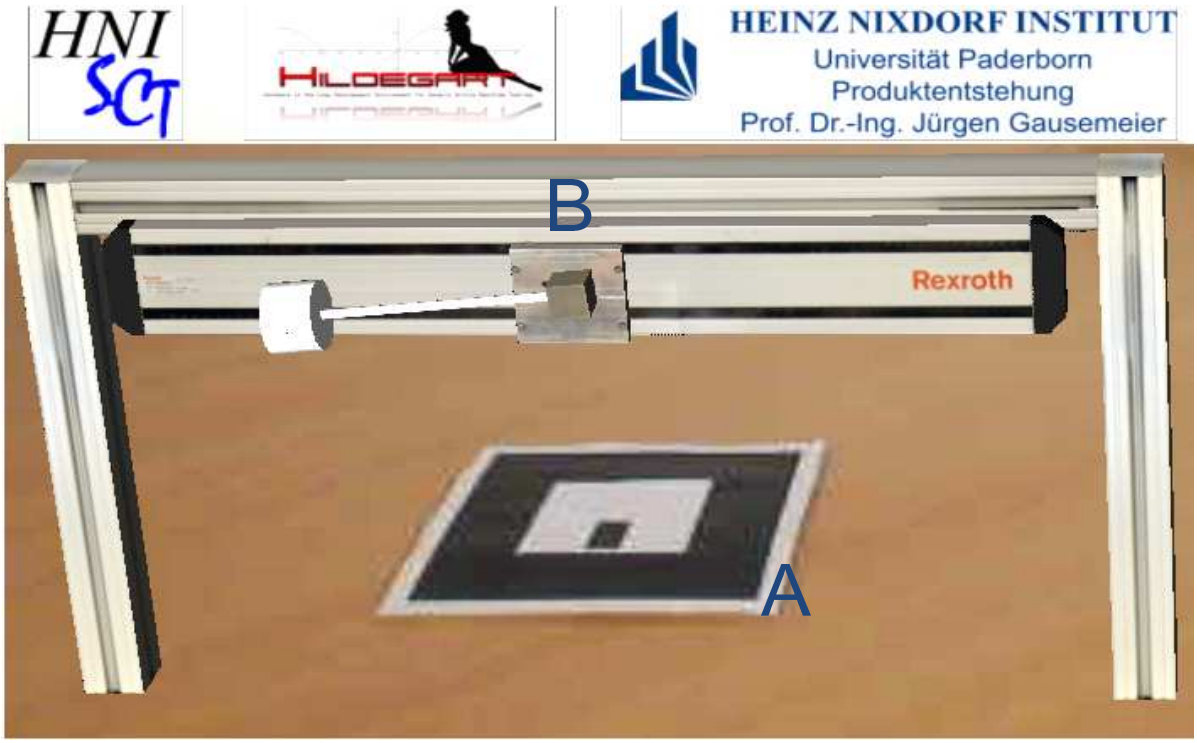


Figure IV. 26: Picture of the PC screen that shows the virtual image of the pendulum (B) moving and its associated pattern (A)

The results obtained in this HiLDEGART test are somewhat discouraging because our pendulum model does not achieve the equilibrium at the inverted position, as we expected based on the HiLDE results. The problem is in the switch between controllers because the angular position of our pendulum model reaches the inverted position without problem, but it can not be balanced. This problem can explain why in the HiLDE test is needed a quarter of second to stabilize the pendulum in the inverted position; thereby, we can dismiss the idea of a timing behaviour problem because of no real time. Probably, the causes of this problem in the controllers switch is because of our parameter identification process and it will be further explained in next section. However, there are some positive aspects to emphasize in the results obtained in this test:

- The first one is that the Augmented Reality is fully compatible with our hardware model and even could say that these two applications are the perfect partner, because both together will be a virtual replacement for the real pendulum.

- The second one is that the Swing Up Controller works perfectly, it has a similar response to the HiLDE test and the pendulum reaches the target angle without problem.

- The third one is that even though the Balance Controller has not been put into operation in this test, there is no reason to believe that it does not have a similar behaviour to the HiLDE test (c.f. Figure IV.21), based on the behaviour of the Swing Up Controller at this test.

- And the final one is that our hardware model of the SCT inverted pendulum can be successfully used for testing the controllers separately in real time and through Augmented Reality.

4.7. Final analysis and possible improvements in our FPGA-based model.

Analysing all the results obtained we can assert that the fundamental improvement must be focus on the characterization of the SCT inverted pendulum, specifically, on the parameters identification process of our non linear grey box model for the pendulum.

Due to the way we have done the parameters identification process, where the final results have been obtained focusing on the swing up response of our model (c.f. 4. Testing with real Swing Up Controller signal in the second chapter). The final parameters found do not work as we expected when the Balance Controller takes the lead and it provides its noisy voltage signal to our model. This problem could be observed in Figure II.42 from the ninth second of the simulation, when the switch between controllers occurs (c.f. Figure II.40), and the result for the cart position moves away from the result measured more than 4 *cm*.

Furthermore, as we have demonstrated with the tests using chirp signals, when the input signal used as training data in order to identify the model parameters has high frequencies the fitting process do not work very well (remember Figure II.38 and II.39 where the results for low frequency variation with time are better than for high).

Thereby, we propose, as improvement, to do a new parameters identification process focus on the balance response of our model, but without forgetting the swing up. Furthermore, this new process will not be as laborious as that done in this master thesis because the starting point can be the test using controller signals as training data and the exponential friction model used in [14].

The last advice for this new parameters identification process is to select carefully and in the smallest possible number the *free* parameters, because as we has demonstrated selecting too many degrees of freedom avoids to obtain a suitable result of identification (remember the test using chirp input signal 1.c and 1.d).

We believe that improving the fitting of our SCT inverted pendulum model, especially in the balance response, the problem in the switch between controllers detected in the HiLDE test and corroborated in the HiLDEGART test will be solved and the pendulum could be balanced as we expected. Moreover, the future changes can be easily included in our project because the only thing we have to do is to set the new parameter values into the Matlab script where the model constants are defined.

Another possible improvement consists on modelling the mechanic subsystem of the SCT inverted pendulum, which consists of the Power Driver Controller, the Motor AC, the Timing Belt and the Spindle block, using a non linear model (c.f. 2.2. section). We believe that not every problem is because of the parameters identification process, the simplification made at this subsystem is also a cause of the differences between our model and the real system. Probably, modelling the mechanic subsystem using a non linear model can help to the new parameters identification process, the drawback is that the number of *free* parameters in our model will be increased.

5. CONCLUSION/OUTLOOK

Concluding this master thesis we want to stress that although the FPGA-Based model of the SCT inverted pendulum does not match exactly with the real system the most of the goals that were set at the beginning of this document have been achieved.

We can assert that a realistic model of the SCT inverted pendulum has been found. Although one subsystem (the mechanic subsystem) has been modelled in a simplified way, the rest (the pendulum and the encoders) have been modelled as realistically and accurately as possible.

In fact, we want to emphasize that in contrast to our approach, the majority of the consulted literature neglect the friction forces. We have considered the Static and the Dynamic (or Coulomb) friction forces, besides the usual Viscous friction, for modelling the pendulum block. Our initial friction model was changed when we realized that it introduced discontinuities that generated problems for solving the differential equations. Then a continuously differentiable friction model replaced the old and discontinuously model obtaining a complete and rather accurate pendulum model, which includes the Static, Coulomb, Viscous and Stribeck friction.

Another important and remarkable point of this master thesis is the encoders model, which has been modelled in a rather novel and particular manner thinking in its final hardware implementation (remember the hardware problem of the encoder model reported by J. J. Incze et al. [23] explained in section 3.4.2. Results of Simulink software model of encoders). In our opinion, this was one of the most realistic ways to model an incremental rotary encoder because the FSM represents perfectly the two possible states of the set LED/photodiode that generates the TTL outputs: Lightness, when a transparent band is between the LED and the photodiode and thereby the output is on high level; or Darkness, when a opaque band is between the set LED/photodiode and then the output is on low level.

Furthermore, this way of modelling the encoders, unlike others that spends a fix period of time counting bands and then generating the train of pulses for the output signal (i.e. high delay), generates the output signal in real time, almost without delay, as the real encoders. Note that because of the optical system inside the real encoders the speed of the output generation is instantaneous (speed of light).

Ending the remarkable point of our encoder model, we want to emphasize its accuracy because the difference between the real position and the position given by our model is tiny, at least three orders of magnitude less, thereby its error is perfectly negligible.

Other interesting point of this master thesis is that some parts of our hardware model, which are not in the Xilinx System Generator library (e.g. the integrators, the absolute value, the multiplier by sign or the saturation block), are reusable and they can be used in future applications of the SCT department. And for sure, our particular software or hardware model of the encoders too.

We also want to emphasize the positive result obtained using the Augmented Reality with our hardware model. Both applications are totally compatible and we have a virtual replacement for the real pendulum due to this perfect partner.

In this outlook section it is necessary to remark the disadvantages that we have encountered during the realization of this master thesis. The main problem was the system characterization, specifically, the parameters identification process, where was impossible for us to find out the exact values of our model parameters. After a great effort, we found an acceptable parameter values that make our model resemble to the real pendulum, but it cannot be a faithfully substitute.

The main differences between the real and our pendulum model are shown clearly when the voltage signal provided as system input is given by the Balance Controller because this high frequency and noisy signal moves our virtual cart in a different way that the real cart. As was explained previously, it is because of the difficulty of the parameters identification process, which was based on the swing up training data. The problem is that due to these differences the HiLDE test has problems to balance our pendulum model and final test, the HiLDEGART execution, cannot balance it.

However, it should not detract from the work done in this thesis because, although we cannot find the exact hardware model for the SCT inverted pendulum, we:

- Have found (a bit different) an inverted pendulum model using a non linear grey box model. This part was complicated because modelling the mechanical part of our system requires mechatronics knowledges, specially, the Power Driver Controller block. The analytical model of the pendulum, which was obtained using differential equations, was, perhaps, the easiest part although considering a complete (i.e. not a simple viscous) friction model turned it complicated. The model of the encoders was a little challenge because there was not literature when it was developed and it had to work in hardware. The calculation of the model parameter required mechatronics and physics knowledges but it was entertaining, particularly, the experimental part for finding out the values of the friction coefficients. And finally, the hardest part of this master thesis because it required the great majority of effort, the parameter identification, which is tricky as Jin et al. [19] said referring to a non-linear model identification using a greybox model because the search for a good fit using training data tend to lead to a increasingly complex model. After months spending on this identification parameters we can assert they were right.

- Have done a Matlab Simulink representation of the inverted pendulum model. This part was easy at the beginning because it consists on translating the mathematical language into Simulink, but we had some problems, e.g. how can we represent differential equation or how can we keep the angular position between -2π and 2π. The hardest task was to design the encoder models in Simulink because some algebraic loops appeared in the early encoder model schematics. We also were worried about the zero-crossing problem at the cart velocity signal, which is the key parameter for selecting between static and dynamic friction force and these zero crossing produce many swapping between them. Finally it was not the cause of the differences between the real pendulum and our software model.

- Have converted the software model into a hardware model. At the beginning we thought that this step will be easy because it should be a simple replacement of the Simulink blocks by Xilinx System Generator blocks, but as we realized some blocks were equivalents but others needed engineering solutions to find out the equivalence, including the encoder blocks, which needed an m-file that described the FSM behaviour instead of the StateFlow Chart used in our software model. Another laborious task was to define every wire and block in our hardware model in fixed-point notation that can represent its value without overflow (i.e. can represent its maximum absolute value).

- Have generated the bitstream and we have programmed the FPGA with our pendulum model. This point of the master thesis was very quick due to the Xilinx System Generator toolbox of Simulink, the V-MAGIC [25] software and the Xilinx ISE Project Navigator. In few minutes we had the bitstream generated and downloaded into the Xilinx Virtex II Pro FPGA of the pendulum module, which is placed adjoining of Controllers module in the RAPTOR 2000 motherboard. And, after connecting the pendulum module to the digital I/O interface of Controllers module, our application was ready to test.

- Finally, we have done several tests. This part required a big effort because we had to analyze many test results (as software as hardware models) to understand the operation of our models and find out the causes that motivate their behaviours in order to improve their results and reach our goals. We believe that every problem found and decision taken has been well argued and explained in this master thesis documentation. And, thereby, the working method and our line of reasoning throughout this master thesis are clear enough.

Therefore, the FPGA-Based Model of an Inverted Pendulum for Hardware-in-the-Loop Simulations has been developed. Although it is not the exact model of the SCT inverted pendulum, the rest of the objectives have been achieved. It is an ideal tool (i.e. smaller, cheaper and safer than a real test bench) for testing controllers separately in HiLDE or HiLDEGART, which can use the Augmented Reality without problem. And also it can be very useful for the SCT department because this model of the inverted pendulum system can be used for designing better controllers. Remember that the current controllers are based on a standard and less accurate pendulum model (without friction). Perhaps, the future Balance Controller will provide a better voltage signal (without noise) that facilitates the identification of the model parameters. Incidentally, remember that future changes can be easily included in our project setting the new parameter values into the Matlab script where the model constants are defined.

6. REFERENCES

[1] Paiz, C.; Porrmann, M.: *The Utilization of Reconfigurable Hardware to Implement Digital Controllers: a Review*. In: IEEE International Simposium on Industrial Electronics (ISIE), June 4-7, 2007, Vigo, Spain, IEEE Industrial Electronics Society, 2007, pp. 2380-2385

[2] Hagemeyer, J.; Kettelhoit, B.; Koester, M.; Porrmann, M.: *A design methodology for communication infrastructures on partially reconfigurable FPGAs*. In Proceedings of the 17th International Conference on Field Programmable Logic and Applications (FPL '07), pages 331-338. IEEE Computer Society, August 27-29 2007.

[3] Paiz, C.; Hagemeyer, J.; Pohl, C.; Porrmann, M.; Rückert, U.; Schulz, B.; Peters, W.; Böcker, J. *FPGA-Based Realization of Self-Optimizing Drive-Controllers.* In Proceedings of the 35th Annual Conference of the IEEE Industrial Electronics Society (IECON 2009), Porto, Portugal, November 3-5, 2009, accepted for publication.

[4] Paiz, C.; Pohl, C.; Radkowski, R.; Hagemeyer, J.; Porrmann, M.: *FPGA-in-the-Loop-Simulations for Dynamically Reconfigurable Applications.* The 2009 International Conference on Field-Programmable Technology (FPT'09), The University of New South Wales, Sydney, Australia, 9-11 December 2009, accepted for publication.

[5] Bosch Rexroth AG: *Rexroth EcoDrive Cs Drives*- Project Planning Manual. Document Number: 120-1000-B344-02. Last change 2004.11.29.

[6] Bosch Rexroth AG: *Compact Modules with ball screw drive and toothed belt drive* - Catalog. Document Number: R310 2602. Last change 2007.02.01.

[7] Bosch Rexroth AG*: Rexroth ECODRIVE Cs Drive Controllers* - Functional Description: MGP 01VRS. Document Number: 120-1000-B355-01/EN. Last change 2003.07.11.

[8] SIMPACK AG.: *SIMPACK Kinematics and Dynamics*. <http://www.simpack.com/kinematics_and_dynamics.html> [June 2009]

[9] Muskinja, N.; Tovornik, B.: *Swinging Up and Stabilization of a Real Inverted Pendulum*. IEEE Transactions on Industrial Electronics, Vol. 53, No. 2, April 2006.

[10] Landry, M.; Campbell, S. A.; Morris, K.; Aguilar, C. O.: *Dynamics of an Inverted Pendulum with Delayed Feedback Control*. In SIAM J. Applied Dynamical Systems, Vol. 4, No. 2, pp. 333-351. 2005.

[11] Stimac, A. K.: *Standup and Stabilization of the Inverted Pendulum*. MIT Bachelor thesis. June 1999

[12] Han, Y.; Tzoneva, R.; Behardien, S.: *MATLAB, LabVIEW and FPGA Linear Control of an Inverted Pendulum*. In AFRICON 2007.

[13] Guillermo, E.; Larriva, J.; Trelles, J.: *Control de un Péndulo Invertido*. 2003

[14] Campbell, S. A.; Crawford, S.; Morris, K.: *Friction and the inverted pendulum stabilization problem*. In "Journal of Dynamic Systems, Measurement, and Control", vol. 130. 2008.

[15] Franco, A.: *Física con ordenador* - Cálculo de momentos de inercia. 1998-2006. <http://www.sc.ehu.es/sbweb/fisica/solido/din_rotacion/inercia/inercia.htm> [August 2009].

[16] The MathWorks, Inc.: *Choosing a Solver* - R2010b Documentation - Simulink - User's Guide - Simulating Dynamic Systems - Running Simulations - Choosing a Solver Type. 1984-2010. <http://www.mathworks.com/access/helpdesk/help/toolbox/simulink/ug/f11-69449.html#f11-41861> [September 2009]

[17] Wikipedia: *Levenberg-Marquardt algorithm*. <http://en.wikipedia.org/wiki/Levenberg%E2%80%93Marquardt_algorithm> [September 2009]

[18] Bun, M.: *Applications of the Levenberg-Marquardt Algorithm to the Inverse Problem*. 2009

[19] Jin, G.; Sain, M. K.; Pham, K. D.; Spencer, B. F.; Ramallo, J. C.: *Modeling MR-Dampers: A Nonlinear Blackbox Approach*. In Proceedings of the American Control Conference. June 2001.

[20] Makkar, C.; Dixon, W. E.; Sawyer, W. G.; Hu, G.: *A New Continuously Differentiable Friction Model for Control System Design*. In International Conference on Advanced Intelligent Mechatronics. July 2005.

[21] Controllab Products B.V.: *20-sim* - Support - 20-sim Help Files - Modeling Tutorial - Friction - Static and Dynamic Phenomena. <http://www.20sim.com/webhelp/modeling_tutorial/friction/staticdynamicphenomena.htm> [October 2009]

[22] Oh, P.: *MEM 351 Dynamic Systems Laboratory 4: Simulink – Modeling Dynamic Systems*. <http://prism2.mem.drexel.edu/~billgreen/mem351/lecture05/mem351Lab-MatlabSimulink.pdf> [November 2009]

[23] Incze, J. J.; Szabó, C.; Imecs, M.: *Modeling and Simulation of an Incremental Encoder Used in Electrical Drives*. In Proceedings of 10th International Symposium of Hungarian Researchers on Computational Intelligence and Informatics, pp. 97-109. November 2009.

[24] Xilinx: *System Generator for DSP* - Products & Services - Design Tools - System Generator for DSP. <http://www.xilinx.com/tools/sysgen.htm> [November 2009]

[25] Pohl, C.; Paiz, C.; Porrmann, M.: *vMAGIC - Automatic Code Generation for VHDL*. In International Journal of Reconfigurable Computing, Volume 2009, Article ID 205149, 9 pages.2009